A DIRECT DIGITAL CONTROL STEPPING MOTOR BUFFER

A THESIS

Presented to

The Faculty of the Graduate Division

by

Kenneth Joseph Ayala

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

Georgia Institute of Technology

August, 1966

In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

3/17/65
b

A DIRECT DIGITAL CONTROL

STEPPING MOTOR BUFFER

Approved: _____

_____

Chairman

_____

_____

Date approved by Chairman: 9/7/66

ACKNOWLEDGMENTS

I wish to take this opportunity to thank my advisor, Dr. John B. Peatman, for his help in identifying the problem and for his aid and advice during the project. Special thanks are also due to Mr. Thomas Marmor for technical help on a critical item of circuitry, and to Mr. J. G. Barnett and Miss Ann Geer for keeping the lines of supply open.

I would also like to recognize the help of Mr. Larry Klingler, Technical Director of Systems Engineering Laboratories, for advice regarding input-output formats for the buffer, and for providing the card carrier used in its construction. Finally, appreciation is due the National Science Foundation for making available funds for the purchase of equipment and the preparation of this thesis, through Research Initiation Grant Number GK263 "Time-Oriented Digital Systems Design."

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS CONTINUED

SUMMARY

The application of Direct Digital Computer Control to industrial manufacturing processes consisting of many controlled sub-loops requires minimization of computer time spent in servicing input-output demands in order that economic use be made of total available computation time. The output information of the control computer, which is to be applied to control the operation of the process, is usually in pulsed-voltage, coded form, representing actions that final electro-mechanical control elements of each sub-loop are to perform. The relatively slow reaction time of the final control elements requires that the control information be stored and presented until the desired action has been accomplished. If the final control element uses a format of information different from that generated by the control computer, the output information from the computer must be modified to suit that required by the final control element.

Stepping motors, a form of multi-pole synchronous motors, represent an actuator which will rotate in one-step increments of rotation when the field coils are energized in a specific order. These motors, used as initial control elements for the more powerful final process control elements such as pneumatic valves, show promise of becoming predominant in Direct Digital Control applications.

The object of this research was to design and construct a stepping motor buffer which is to be interposed between one output channel of a

control computer and 63 stepping motors. The input to the buffer is a 17 bit binary word which expresses, in binary-coded form, the address of the motor to which the word is directed, the number of steps to be taken by that motor, and the direction of rotation. The input words may be transferred at an average rate of one every millisecond, or one every two milliseconds in the worst case. The output of the buffer is 63 channels, one to each motor, over which the proper pattern of pulses, which energize the field coils, are sent. Each stepping motor is considered to have a maximum step rate of 100 steps/second; thus the stepping commands are sent from the buffer to the motors at a rate of one every ten milliseconds for each motor being stepped.

The buffer was designed and constructed using a commercially available make of integrated-circuit resistor transistor Logic, and uses a sonic delay line as a memory element. Logic elements are of three types: NAND gates, JK flip-flops, and fan-out buffers. The stepping motor buffer was designed to accept and impart information at the rates given above, process the input format to a form suitable for application to the stepping motors, and to direct the appropriate information to the desired stepping motor. The buffer will update any information stored in it by adding the incoming information to that already stored.

The buffer was tested with one stepping motor as a representative output, and a set of toggle switches were used to simulate the input word from the computer. Tests were made to ensure that the motor would step the exact number of steps set in for its address, that it would step only when addressed, and that the buffer could be updated.

The results of the tests demonstrated that the buffer met all requirements.

CHAPTER I

DIRECT DIGITAL CONTROL

Evolution of Direct Digital Control (DDC)

Man's interest in the control of dynamic systems may date from the time the first rock was directed toward the head of a game animal or an enemy. Since that time the pursuit of more accurate, faster-responding and cheaper control systems, whether for material benefit or defense needs, has occupied the thinking of a large segment of the technical community.

In this century the occurrence of two world wars and the rapid growth of scientific and engineering knowledge has evolved control systems which find application in such diverse areas as bio-electronics, missile guidance and automatic cake baking. One of the most prolific users of control technology is the material processing industry, particularly the petro-chemical division. To economically process and derive the numerous products called for in a chemical process demands the construction of large processing systems and correspondingly sophisticated means of controlling the diverse parts of such a system. Current practice, in a typical process application involving many inter-related loops, utilizes discrete controllers for each process loop, as shown in Figure 1.[1] Here the loop variables are measured and compared with a set point reference, and the controller generates

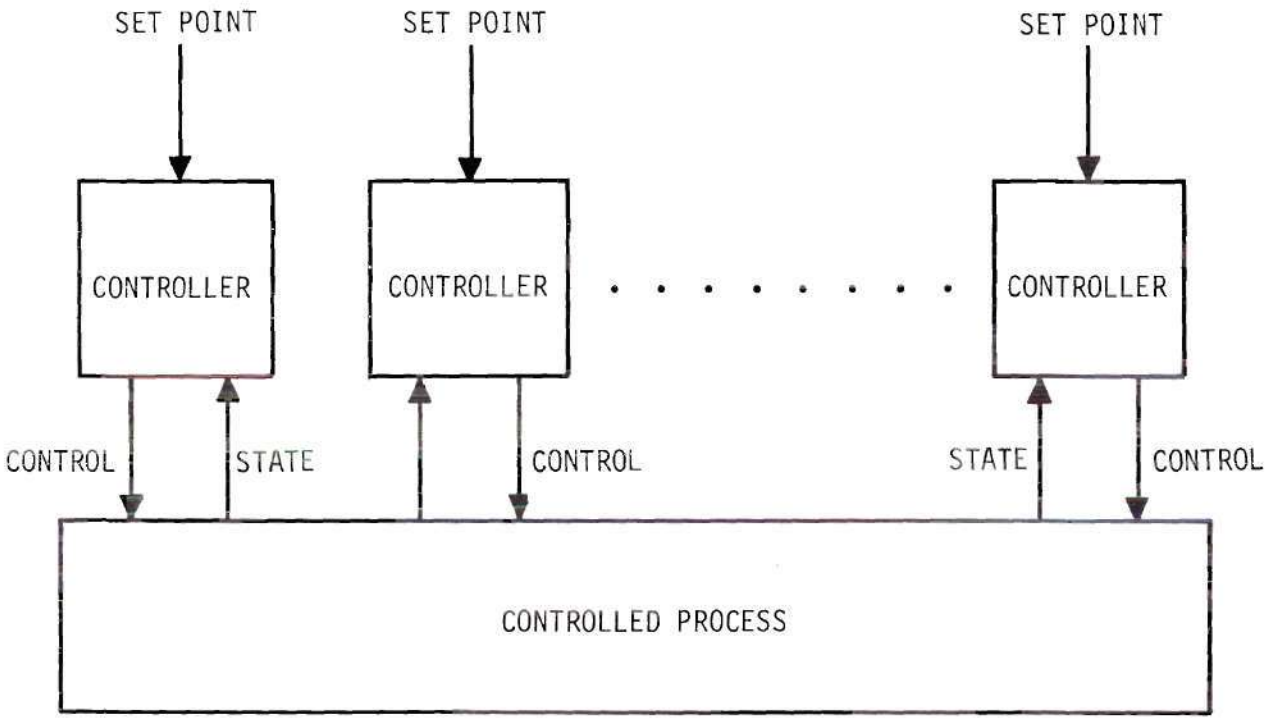


Figure 1. Discrete Loop Control.

appropriate control signals to drive the final control elements. The set point is established external to the loop in question and may be determined by the action of another loop, or as a result of operator calculation. The majority of the controllers are analog devices, thus representing a fixed control system which does not lend itself to extensive system modification or experimentation. Furthermore, such devices are subject to drift and inadvertent operator interference. While these systems have proved to be very satisfactory in process control applications, the need for even greater control accuracy, particularly as process systems grow in size and instability, has forced the industry to consider new device application.

The success of the aerospace industry in utilizing digital devices for the control of high-performance aircraft and missiles led process engineers to consider the digital computer as a direct control element. Prior to this time the digital computer had been primarily used for data processing and problem solving on a non-real-time basis. The high cost and relatively poor reliability of the digital computer precluded its application to real-time process control at that time, but advances in the field pointed toward overcoming these difficulties. In 1956, engineers of the DuPont Company, after initial experiments, determined that a centralized digital computer could be used to replace the individual controllers of a multi-loop system.[1] This study encouraged other companies to begin investigation of the digital computer as a control device. In the late 1950's several installations of a hybrid sort were made, with a control computer calculating the set points for the conventional controllers and thus

acting as a system supervisor.[2] By 1959 The Imperial Chemical Industries Company, a British firm, began planning for the introduction of a digital computer to control a 98 loop ammonia plant, and, in 1962, the computer, a Ferranti Argus 200, was on line in the plant.[3] Simultaneously Monsanto installed a computer control system in its Texas City plant.[1]

The unique factor in both of these applications is that the computer is used to replace the individual loop controllers, and the final control elements are directly operated by the computer. The Direct Digital Control of a process then takes the block form of Figure 2. Here the computer is time shared between the various loops at such a rate that the loop, as a user, believes it has the complete attention of the computer. The loop variables are measured and read into the computer, where calculations based on these variables, the control equation and a stored set point are done. Suitable command instructions are then evolved and applied to the process final control elements.

The concept, then, of DDC is initially the replacement of many conventional controllers with one digital control computer; however, a better definition might be: the control of any system by the use of a time-shared digital device which receives information on the state of the system directly, and evolves corrective signals which are applied directly to the final control elements. Moreover, the computer may also engage in optimization of the system operation, data logging, alarm indication, visual displays of operations status and many other desirable services.

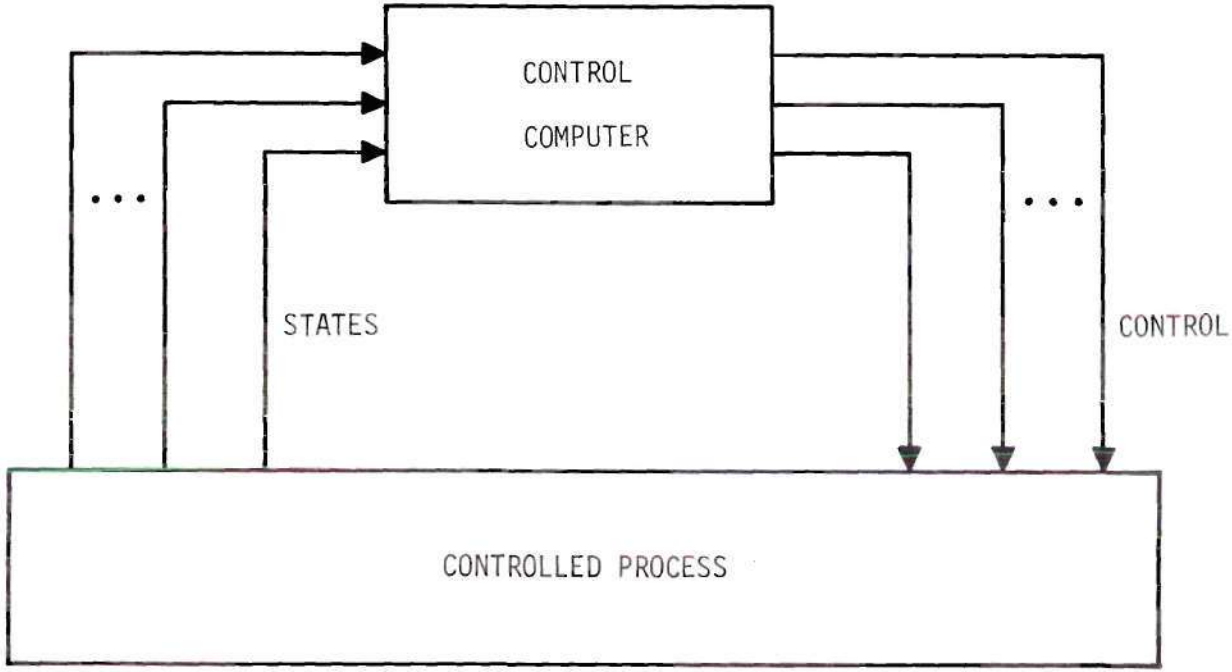


Figure 2. Direct Digital Control.

The success of the early systems generated active interest by other users and vendors of computer equipment. Currently 20 known applications of DDC in industry have been found, and 21 vendors are marketing process control computers.[4] As an example of the rapid application of DDC, the Esso Company is installing a Foxboro DDC System in their Fawley, England refinery. The DDC system will use one master and two slave computers to direct the operation of 400 valves and is due to begin operation in October, 1966.[5]

The advantages of DDC over discrete conventional controllers are many and among these are lower cost (tentatively set for systems with more than 50 loops[1]), faster response, ease of expansion and modification, smoother control, and adaptability to various process systems. If to these are added optimization of plant operation, system status displays, data logging and reduction, and ease of operator control, the realization of a completely automatic plant is at hand. The result of DDC has been increased operation output, lower cost and increased efficiency, and it is thought that some processes would not have been possible without DDC.[6]

Requirements for DDC

By 1963 sufficient experience and interest had been created in DDC for prospective users of DDC to make known their needs to the computer vendors. In April of that year some 20 user companies held a workship at Princeton University under the auspices of the Chemical and Petroleum Industries Division of the Instrument Society of America. As a result of this meeting a set of guidelines was established for vendor use.[7] These guidelines were concerned with classification of

DDC system types, accuracy, reliability, maintenance, safety and input/output requirements. The vendors were then faced with the task of evolving systems to meet these requirements.

Early in the development of DDC engineers foresaw the problem of efficient input/output operation when this classification includes communication between computer and process, computer and computer, and computer and operator.[8] Figure 3 shows, in block diagram form, what may be considered a typical DDC system. The input/output relationships between a central processing computer and the system being controlled by that computer result in certain discontinuities or "interfaces" between the two halves. Commonly, the system transducer elements produce analog outputs which must be converted to digital form for application by the computer, and the computer output commands, usually in digital form must be converted to analog equivalents for application to the analog final control elements. The evolution and introduction of digital transducers and digital control actuators promise to erase this barrier between system and process.

A second barrier exists, however, which does not grow smaller as equipment is improved, and that is time. The central computer usually has the ability to sample the process variables, compute the corrective information, and make this information available to the final control elements at a much faster rate than the control elements can follow. Some means must then be found to store this control information and direct it to the proper actuator at a rate compatible to the actuator. The excess time then available between control calculations may be used to perform other desirable tasks. The storage

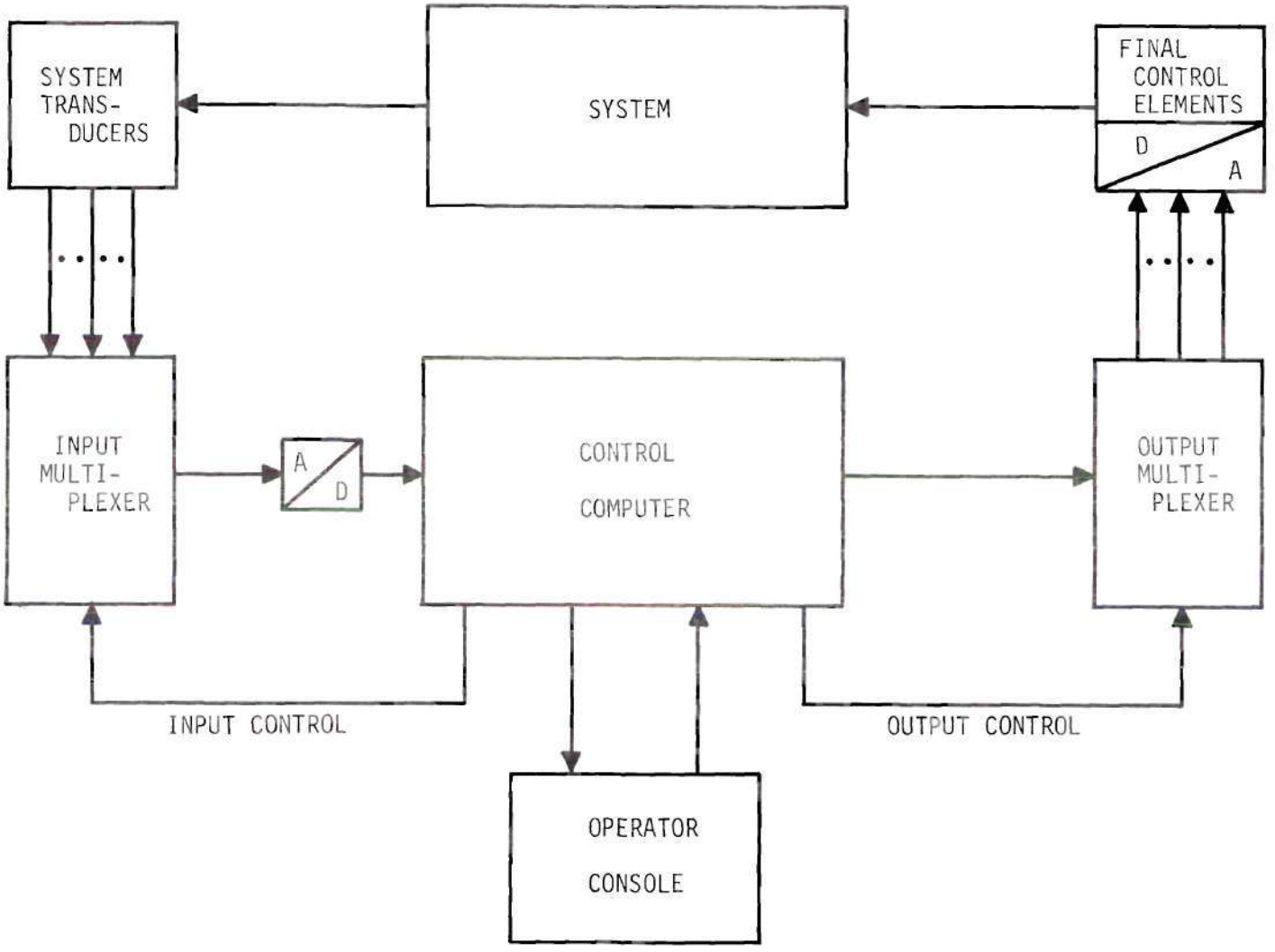


Figure 3. Direct Digital Control System.

and direction of the calculated correction data may generally be accomplished by two means: programming or "software" which utilizes the existing computer equipment and memory capability to hold the information and direct it to the proper actuator, or adding additional equipment or "hardware" to accomplish the same task. Using software to accomplish the objective results in better system reliability because no additional complexity is involved; however, it also results in an increase in non-productive use of computer time for the handling of the information. For DDC applications this may represent an appreciable reduction of computer efficiency when a large number of outputs are to be processed. Adding more hardware to the system will lower system reliability, but will enable the computer to engage in more productive work.

The research conducted in this thesis is directed toward the realization of the second of the two approaches: the design and construction of a fixed logic (wired logic) device which buffers information between the central process computer and a number of control actuators. Generally the input to the buffer consists of a single channel from the process computer, and the output of the buffer consists of the several channels to the control actuators. As used here the buffer has the following characteristics:

1. It accepts information from the central computer at a rate determined by the computer.
2. It uses the latest information to update present contents.
3. It stores the information.
4. It processes the information to obtain a form compatible for

application to the specific control actuator.

5. It directs the information to the proper actuator at a rate which may be followed by the actuator.

The buffer should also meet the requirements of low cost, high reliability (99.95 per cent for DDC), ease of maintenance and operation, and it should be operable in an industrial environment. The buffer which is described in this thesis has been designed to meet these requirements; however, extensive testing to prove the degree of reliability is beyond the scope of the work done. Figure 4 shows the configuration of such a buffer in a DDC system.

The buffer herein designed is meant to be used with a certain type of stepping motor of a general class described in Appendix I. Currently two methods are used to convert digital signals to analog form: digital to analog resistive ladders, and stepping motors. The resistive ladder is the faster acting of the two, converting digital pulses to a voltage level. However the voltage level must then drive an analog actuator and performance is then dependant on the characteristics of the actuator. The stepping motor converts digital pulses into discrete rotational steps at a rate determined by pulse rate inputs and limited by mechanical considerations. The stepping motor, which may be considered an incremental synchronous motor, is a true digital actuator, converting information in discrete digital form into discrete increments of rotation with no error accumulation. As such it holds great promise as a true DDC actuator element, and manufacturers are now designing new final control elements using the stepping motor as the control actuator.[6]

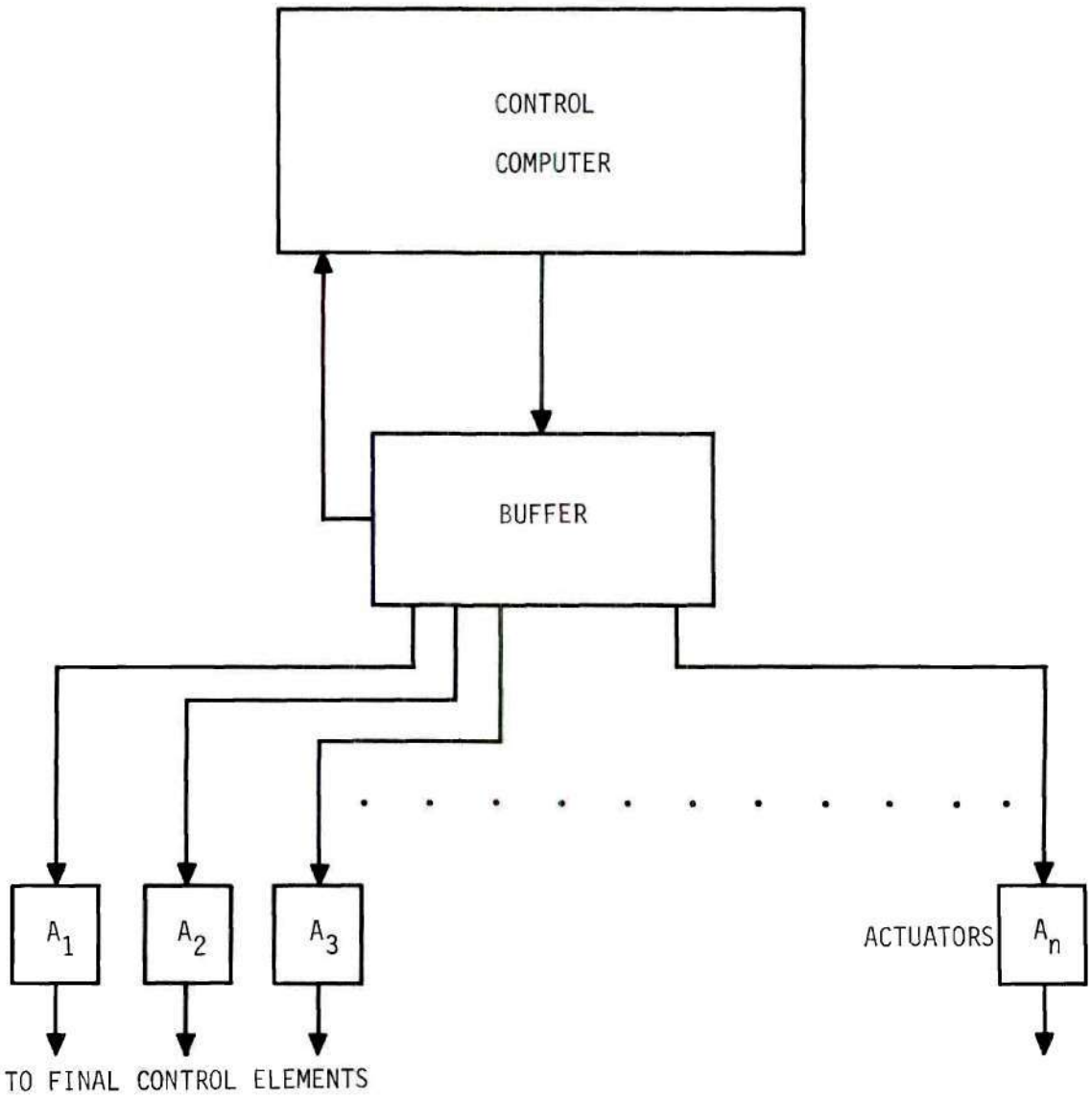


Figure 4. Control System with Buffer.

CHAPTER II

STEPPING MOTOR BUFFER DESIGN

Introduction

The main objective of this research has been to design and construct a prototype of a stepping motor buffer which will meet the general specifications outlined in Chapter I. In this chapter the input and output constraints for the buffer are considered, and, based on these constrainsts, the design of the buffer is presented. First the buffer operation on an over-all basis is described; then the operation is divided into four main phases, or logic blocks, and each is examined in detail. The chapter concludes with a section on construction of the buffer.

Input and Output Constraints

The choice of input word format was dictated in part by the ISA guidelines.[7] The break-even point, where DDC becomes economically competitive with conventional, discrete analog controllers, has been set at 50 loops. Considering each stepping motor to be the control actuator for a single loop final control element, the buffer must be able to direct at least 50 stepping motors. Because no additional logic, and in fact less, is required to direct 63 stepping motors (exclusive of the additional 13 motors and motor drive logic), this was the maximum number of stepping motors selected to be directed by

the buffer. The ISA guidelines also specify that analog-to-digital conversion accuracy is to be .01 per cent or one part in a thousand. This necessitates the same accuracy or resolution for digial-to-analog conversion; therefore the input word magnitude group must have the capability to express any number from zero to 1000. The direction of stepping must be specified, and the simplest method is to assign one bit to signify the direction. The input word is expressed in binary form because this is the most likely, and natural, number system for the computer to utilize. Finally, because it is desirable to minimize computer storage time in the handling of output information, the words are presented in parallel form to the buffer.

The input to the buffer, then, is serial command words, each presented in parallel form, consisting of 17 binary digits or bits. Each digit of the input word is represented by the presence or absence of a specified voltage level on the lines incoming from the computer and is considered to be presented simultaneously. The format of the input word is shown in Figure 5. The input word is divided into three groups of bits: one group of six bits to identify or address the specific stepping motor to which the word pertains; one group of ten bits to express the number of steps the addressed motor is to perform; and one group of one bit, termed the sign bit, to specify the direction, either clockwise or counterclockwise, that the motor is to step.

The requirements of the particular stepping motor used fixes the format of the buffer output. Because all of the stepping motors are considered to be identical, it is only necessary to investigate the characteristics of a typical motor. The stepping motor used was a

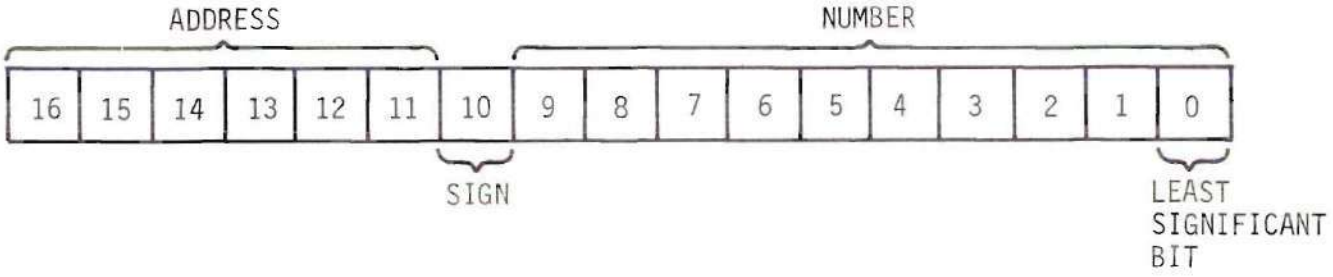


Figure 5. Input Word Format.

bifilar-wound, permanent magnet type, as shown schematically in Figure 6 (See also Appendix I). The motor may be made to step in a desired direction by the application of a sequence of voltage levels to the four field coil leads. Refering to Figure 6, and numbering the field coil leads from one to four, as shown, the motor will step one step for each sequence of voltage levels impressed on the coils. Table 1 gives the voltage sequences for rotation of the motor in each direction (+ means the coil is energized, and - means the coil is de-energized).

By assigning logical one to plus and logical zero to minus and noting that coils one and three are the logical complement of each other, as are two and four, the sequences given in Table 1 may be reduced to the sequences shown in Table 2. It is understood that coils three and four are connected to the complement of coils one and two respectively. This logical sequence, which is commonly known as a two-bit Gray code, is the output of the buffer to each stepping motor. It should be noted that to step in one direction a given sequence is used, and to step in the opposite direction the reverse of the original sequence is used. The rate of stepping is determined by the sequence rate. Between sequences, or steps, the buffer must apply to the field leads the last sequence generated, so that the motor will always be properly oriented.

Operation of the Buffer

In Chapter I the general characteristics of the buffer were given. Based on the input/output constrainsts of the previous section, the characteristics of the stepping motor buffer may be more fully specified. They are:

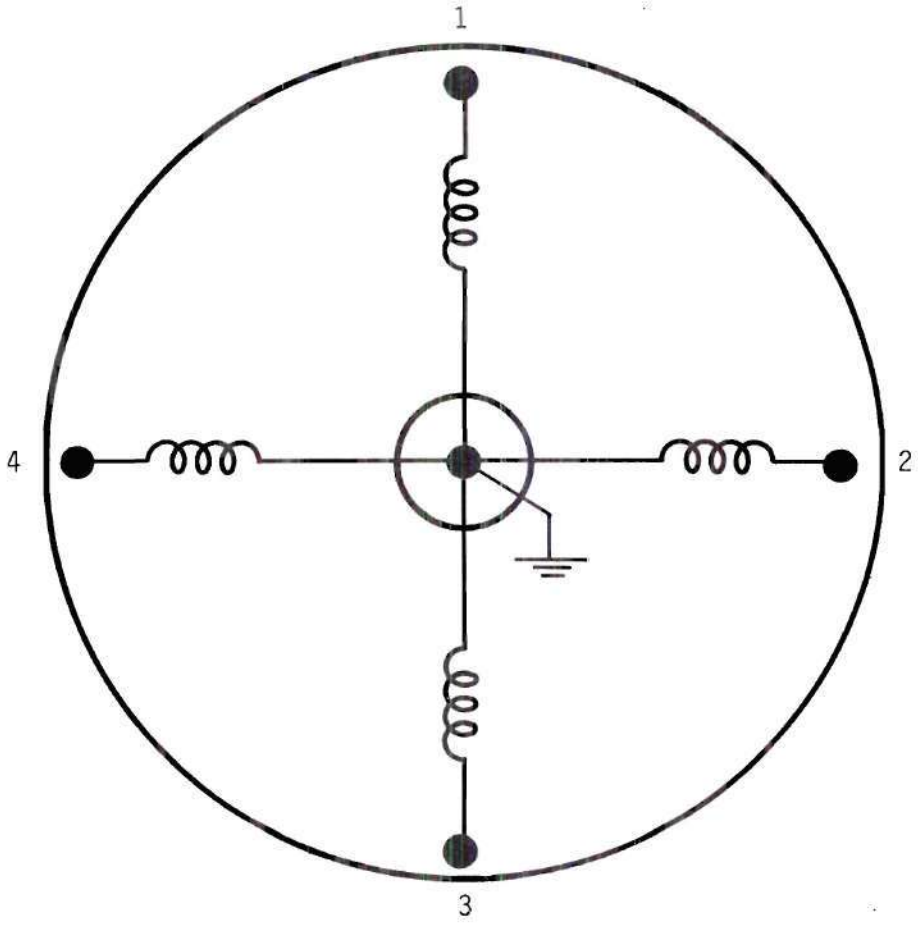


Figure 6. Schematic Diagram of Stepping Motor Field Windings.

Table 1. Stepping Motor Voltage Sequence

Direction 1				Direction 2			
Coil				Coil			
1	2	3	4	1	2	3	4
+	+	-	-	+	+	-	-
+	-	-	+	-	+	+	-
-	-	+	+	-	-	+	+
-	+	+	-	+	-	-	+
+	+	-	-	+	+	-	-

Table 2. Stepping Motor Bit Sequence

Direction 1		Direction 2	
Coil		Coil	
1	2	1	2
1	1	1	1
1	0	0	1
0	0	0	0
0	1	1	0
1	1	1	1

1. The buffer will accept control words from the computer at a rate determined by the computer up to a maximum, worst case, of one word every two milliseconds (1023/506500 seconds). This restriction is a result of the memory system used, as explained below in the section on the Memory Address Logic Block.
2. The buffer will combine the latest input word with the data currently contained in that word address. The incoming word will be added or subtracted from that present, based on the relative polarity of their sign bits.
3. The buffer will store the information as it is being used.
4. The buffer converts the input data into the proper sequence and number of Gray code pulses for application to the stepping motors.
5. The buffer directs the appropriate Gray code pulses to the selected motors at a rate the motors can follow. The motors used are considered to have a maximum reliable step rate of 100 steps per second; therefore the buffer will step each selected motor at the rate of one output pulse sequence every ten milliseconds (5120/506500 seconds) until the specified number of steps have been accomplished.

The block diagram of Figure 7 shows the stepping motor buffer divided into four areas of operation, or logic blocks: Input Logic Block, Memory and Address Logic Block, Operate Logic Block, and Motor Output Logic Block. This discussion is intended to describe the function of the various blocks in the operation of the buffer; subsequent sections will describe the logical design of the logic blocks themselves.

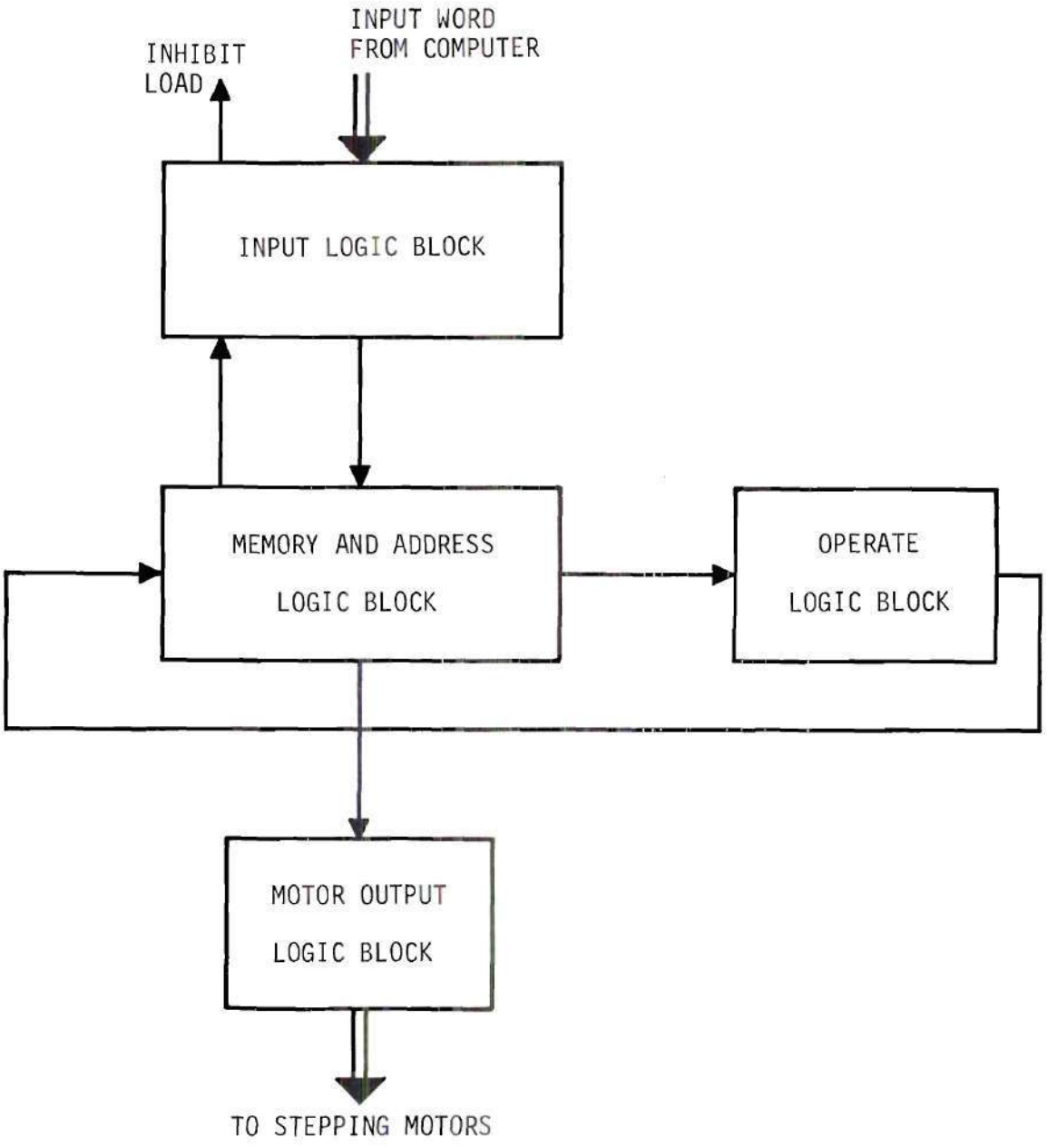


Figure 7. Block Diagram of Buffer.

The buffer is a wired-logic device; that is, it will perform logical operations, based on the manner in which the basic logical circuit elements are connected and on any initial conditions or logical states which exist at an arbitrary time reference. The logical operations that are carried out are based on a plan of calculation, or an algorithm, which accomplishes the desired objective. The objective of buffer logical operation is to take a signed number expressed in binary form and reduce it to an equivalent two-bit Gray code sequence. By repeating this procedure any quantity of such numbers may be converted. The basic algorithm used here is to decrement the number periodically by one. Upon completion of a decrement operation the new Gray bit sequence is formed, based upon the previous Gray bits and the sign bit of the number. This is repeated until the number is zero. In this manner a number is counted down to zero, and a new Gray bit sequence is formed on each count. By suitable choice of decrement period, the Gray bits may be applied to the stepping motors at a rate which they will reliably follow. The logic blocks, as shown in Figure 7, function in such a way that the basic control algorithm, as well as peripheral operations for input/output requirements, is performed.

Before a description of buffer operation is presented, an explanation of the memory system used is necessary. The buffer must store information, and this is the function of the Memory and Address Logic Block. Memory capability is provided by a sonic delay line which stores information in the form of torsional strain waves in a thin wire. Two milliseconds are required for one bit to travel the length of the line, and this is the minimum memory access time. Information

may be placed in the line and recirculated, thus forming a permanent store for the information. Information is inserted in the memory in synchronism with the pulse of a frequency stable pulse generator, or clock, so that a bit space is considered to be entered in the line for each clock pulse. The frequency of the clock determines how closely spaced are the bit spaces, and, for a fixed length delay line, determines how many spaces are on the line or stored at any time.

The buffer stores information pertaining to 63 separate motors; the total number of bits is divided into groups, or words, each word associated with a specific stepping motor. The address circuitry then assigns to each word or word space on the delay line a number which is the same as a corresponding stepping motor. Within each word the bit position is also assigned a number so that each position may be identified. Figure 8 shows the format of a buffer word. Reading from right to left, the zero bit position is unused, the next ten bits are used to store the number of steps, the eleventh bit is reserved for the sign bit, the twelfth and thirteenth bit positions are for the Gray bits, and the fourteenth and fifteenth spaces are unused. There are 63 such words, and the zero address word which is not used, on the line, making a total of 64 words of 16 bits each, or 1024 bits. The address action specifies that a certain word is available when the first bit of that word is in a position to be clocked into the delay line. Thus the word spaces are in constant circulation in the memory system at a bit rate determined by the clock, and are identified by the address system as they circulate.

The operation of the buffer may now be described, with reference

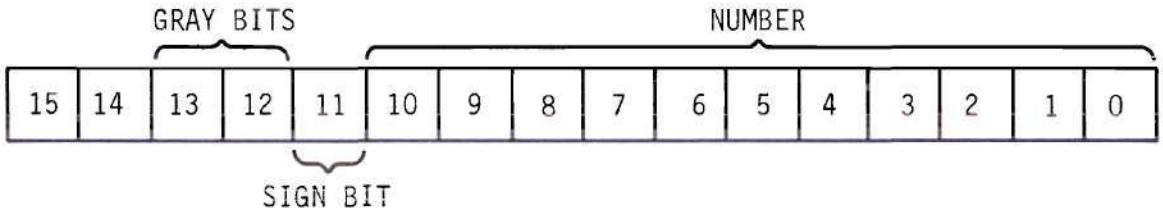


Figure 8. Buffer Word Format.

to Figure 7. Input words from the computer may be loaded into the Input Logic Block unless an Inhibit Load level is present from the Input Logic Block. The purpose of the Inhibit Load level is to prevent the sequential loading of input words at a faster rate than may be handled by the buffer. If the level is present, a previous input word is still in the Input Logic Block, and it is not possible to load in an additional word. After a word is admitted to the Input Logic Block, it is stored until it may be combined with the current contents of the correspondingly addressed word space in the memory. The sequential load-in rate of input words is thus limited because of the two millisecond delay, worst case, in obtaining a word from memory. The total load in time of numerically sequential words may be minimized by noting that once the initial address is located, all sequentially addressed words which follow may be loaded in within two milliseconds.

When the specified word space becomes available, the latest number of steps, as specified in the input word is to be combined with the contents of the number section of the word space. The respective numbers are added if their sign bits are equal and subtracted if the sign bits are unequal. The Input Logic Block determines when the desired word space is available, and the two numbers are combined. The new information is then recirculated in the memory system.

All buffer words are operated on periodically by the Operate Logic Block in order that the basic algorithm be carried out. The Operate Logic Block times the interval between operations so that every ten milliseconds the value of all numbers different from zero

currently in the memory are decremented by one, and new Gray bits are formed. The number and new Gray bits are then stored until the next operational cycle. Once a number reaches zero, the operations stop and the last Gray bit sequence is stored. The Motor Output Logic Block takes the Gray bits that are formed and converts them into voltage levels compatible with the stepping motor requirements and stores the latest Gray bits so that the output levels are maintained between sequences. The selection of the stepping motor to which a particular output applies is also done by this block.

Logic Element Descriptions

The logic blocks, with the exception of the delay line circuitry and motor driver, are implemented using six basic logic elements, shown symbolically in Figure 9. The operation of the logic elements may be defined in terms of certain voltage levels and pulses, shown in Figure 10. The two voltage levels are 0 volts and +3.6 volts, shown in 10(a) and (b); they will be referred to as - and + levels respectively. Pulses are considered to be transitions from + to - and return, as shown in 10(c). The leading edge of the pulse, or the + to - transition shown in 10(d), will be referred to as the alpha (α) transition.

A two-input logic gate is shown in Figure 9(a), with its voltage "truth table." The truth table gives the level of all possible outputs for all possible input combinations. By assigning voltage levels to be logical one or logical zero, independently, for the input and output, the gate may be considered to be a negative-true (input and output) NAND gate, a positive-true (input and output) NOR gate, a

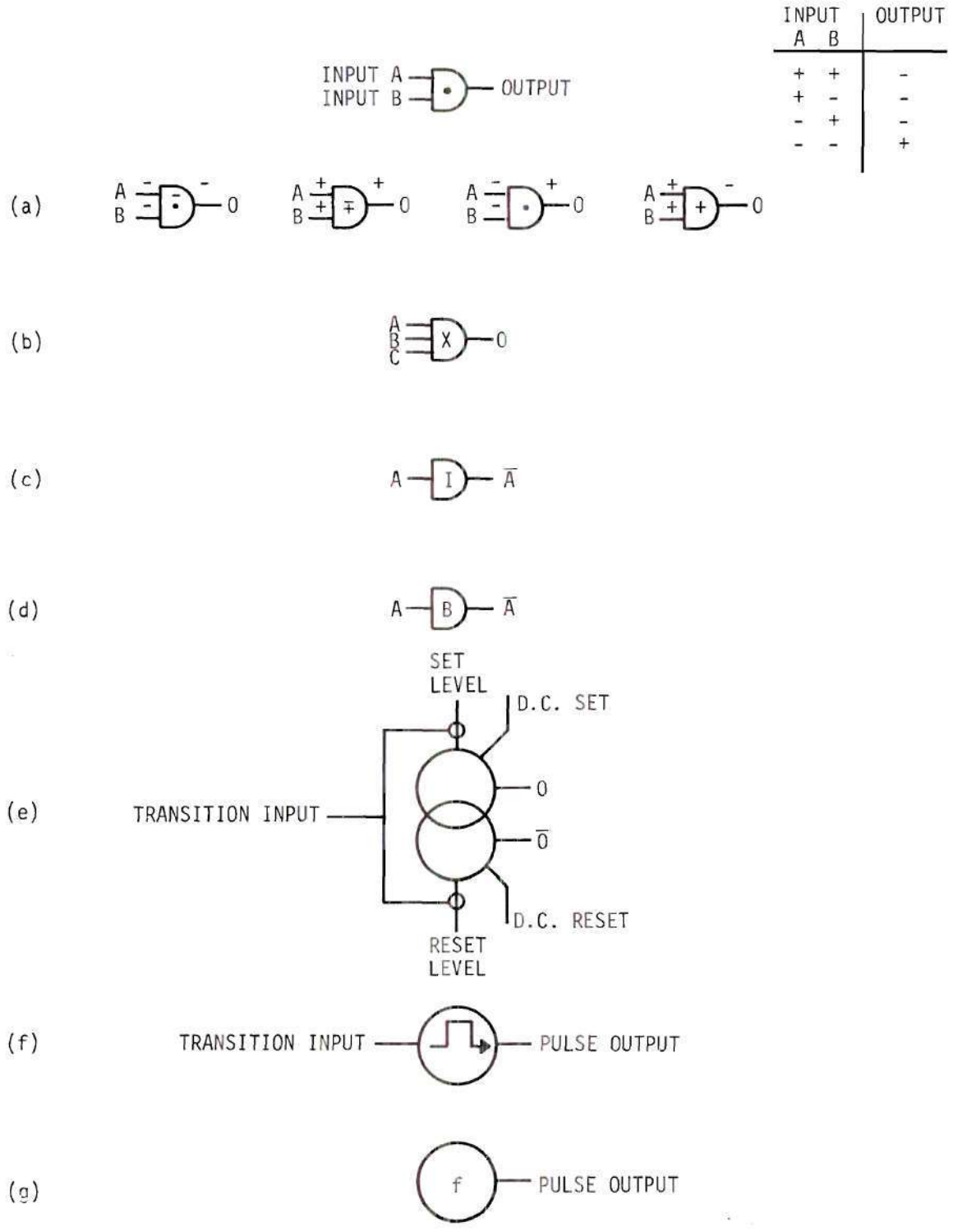


Figure 9. Logic Element Symbols.

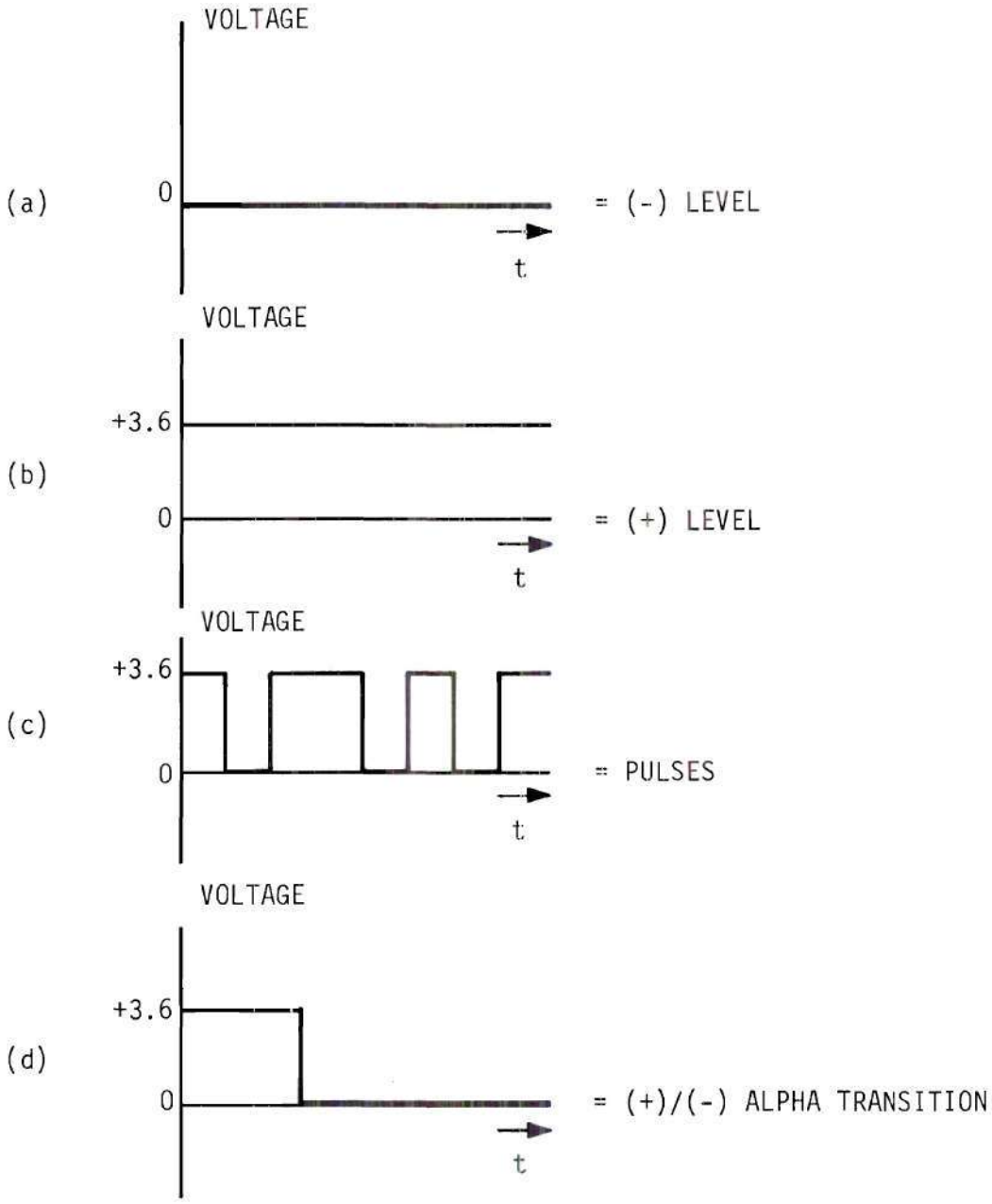


Figure 10. Voltage Levels and Pulses.

negative-true (input), positive-true (output) AND gate, and a positive-true (input), negative-true (output) OR gate. Figure 9(b) shows the symbol for a three-input gate expander, which may be used to increase the number of inputs of a gate by tying the output of one or more of the expanders to the output of the gate. The combination of expanders and gate may then be considered to be a single gate. Figure 9(c) shows the inverter, which is a single input gate and always inverts the level applied to its input. The inverter may be used with the expanders to form a multiple input gate. Figure 9(d) shows a fan-out buffer which has the capability to drive more loads, or inputs, than the other logic elements. Logically it inverts the level applied at its input, but it may not be used with the expander. Figure (9e) shows the symbol for a bi-stable multivibrator, or JK flip-flop. The flip-flop has two outputs, O and $\bar{O}$, which are the logical complements of each other, so that if O is positive, $\bar{O}$ is negative, and vice versa. The outputs of the flip-flop may be made to change states in two ways. If a + level is applied to the D.C. set input, O will go to -, and $\bar{O}$ to +. If a + level is applied to the D.C. reset input, O will go to +, and $\bar{O}$ to -. If both are applied simultaneously both outputs go to -. Once the flip-flop has assumed a new state, the D.C. level which caused the change may be removed. The second method is by means of the level set and reset inputs in conjunction with an alpha transition on the transition input. If a - voltage is applied to the set level input and an alpha transition is applied to the transition input, O will go to -, and $\bar{O}$ to +. If a - level is applied to the reset level input and an alpha transition is applied to transition input, O will be +, and $\bar{O}$ will

go to -. If both levels are - and an alpha transition is applied, the flip-flop will "trigger" or assume the opposite state of that which existed before the alpha transition was applied. The D.C. set and reset inputs will override the level set and reset inputs. In the discussions that follow, a flip-flop input will be said to be "enabled" when a level input is at - and the D.C. levels are at -. Figure 10(f) depicts a "one-shot", an element which will produce an inverse pulse on the output when an alpha transition is received at the input. Figure 10(g) shows the symbol of the pulse generator, or clock, which is the basic timing element of the buffer. The clock does not appear in any logic block and may be considered to be a separate entity. Using these elements, the detailed description of the logic blocks may now be presented.

Memory and Address Logic Block

The heart of the buffer is the Memory and Address Logic Block, which is shown in Figure 11. The input to the memory system is from the Input Logic Block, and the output of the system goes to the Operate Logic Block. The portions of these blocks in series with the memory system consists of combinational logic, and no time delay is associated with them; therefore, the memory system may be considered closed in regard to bit spaces.

The basic storage element, the delay line, is shown in block form, with the delay line driver and reader shown symbolically. The operation of the delay line, as well as the driver and reader, is discussed in Appendix I. When a pulse is fed into the delay line driver, it is converted into a torsional pulse which travels down the line. At

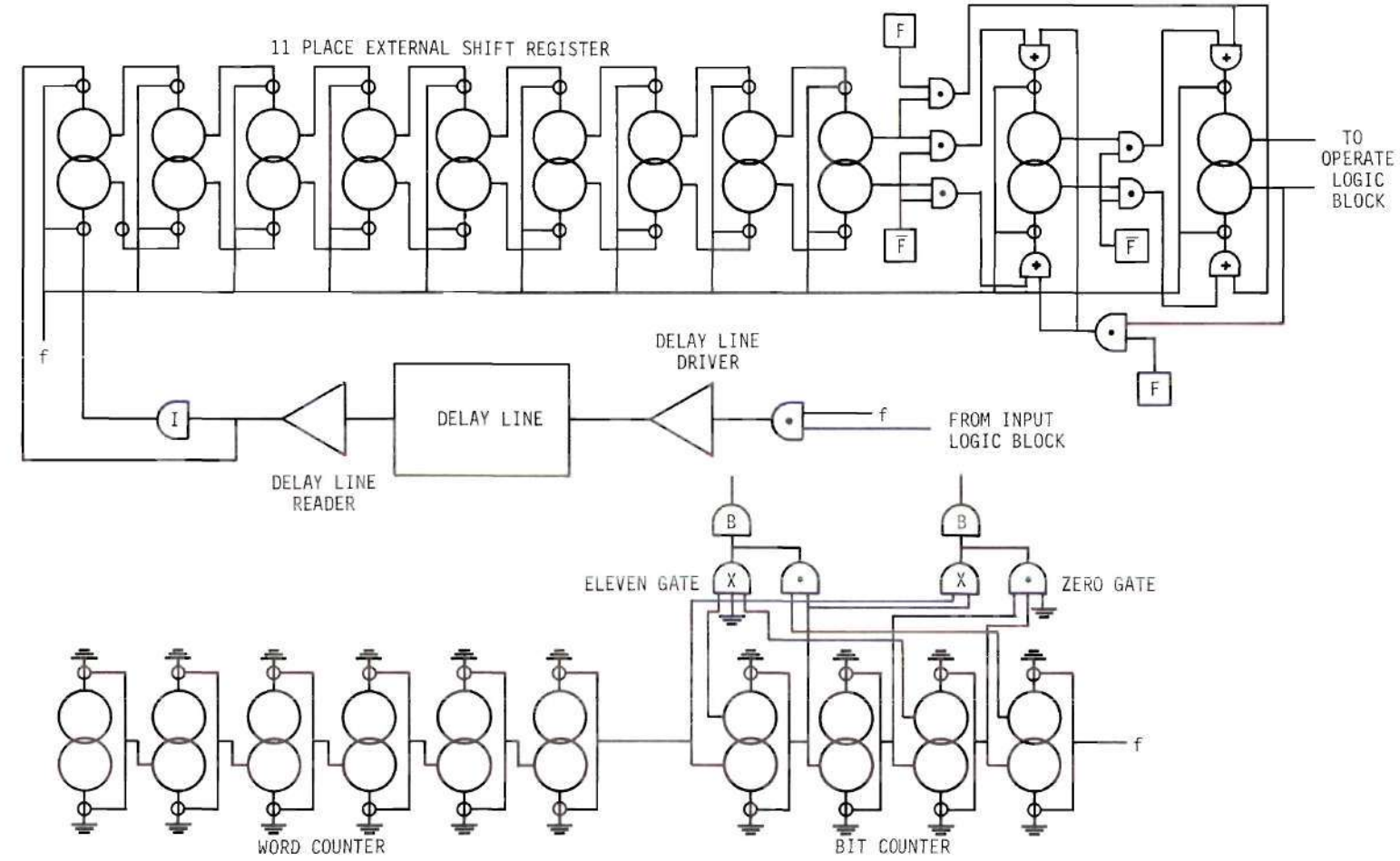


Figure 11. Memory and Address Logic Block.

the end of two milliseconds (± two microseconds, adjustable) the pulse reappears at the output of the delay line reader. The frequency of pulse input will determine how many pulses are stored on the line. This may be expressed as:

$$P = fd \qquad (1)$$

P is the maximum number of pulses stored on the line; f is the frequency of input pulses in pulses per second; and d is the delay of the line, in seconds. Given a fixed length line, or delay, the frequency of the input pulses determine the maximum storage capacity of the delay line. A pulse, or bit of information, that has been put into the line is known to be present at the output of the reader at a later fixed time. A bit may be identified, then, if the time after its insertion into the line is measured. At the end of the delay period the pulse which appears at the output of the reader will be that originally inserted at t = 0. Because information is stored in the delay line at the frequency or clock rate, f, all bits will be circulated through any external circuitry at that frequency.

The main memory storage used is the delay line, but because it is necessary to be able to determine the states of several bits in each buffer word before certain operations take place in the Operate Logic Block, an external circuit is provided which will store the necessary bits. The external circuit is called a shift register and is formed by connecting the outputs of consecutive flip-flops to the set and reset level gates of the flip-flop in front of them, as shown in Figure 11. Any bit stored in one flip-flop will be set into the

succeeding flip-flop when a clock pulse alpha transition is applied to the transition input of the succeeding flip-flop. A chain of such connections may then be built up depending upon how many bits are to be stored in the register. The External Register consists of 11 flip-flops; thus 11 bit positions are added as part of the memory store. Delay lines may be constructed so that they may be tapped along their length, and such a line would obviate the need for the External Register, but would require 11 delay line readers, probably a poor trade-off. The total memory store is now fixed, and the clock rate f is determined by the memory configuration. The buffer is required to store 1024 bits, 11 of which may be stored in the External Register; thus P in equation (1) is 1023. Therefore,

$$f = \frac{P}{d} = \frac{1023}{2 \times 10^{-3}} = 506.5 \times 10^{3} \text{ pulses per second.} \qquad (2)$$

The reliable operation of the delay line requires a stable clock frequency, and a crystal-controlled oscillator is used to generate the clock pulses. (See Appendix I for details on the clock.)

As previously stated, all operations of the external circuitry are synchronized with the clock frequency. Voltage levels which appear on the output of the Input Logic Block are to be stored on the delay line. These levels may change at every alpha transition of the clock pulses, for it is on this transition that the External Register shifts bits along its length and into the external, combinational circuitry. Therefore, at the end of each alpha transition the level presented to the input of the delay line represents a bit. The level is ANDed with

an inverted clock pulse $\bar{f}$, which is opposite the polarity of f. This is done so that the level will have stabilized when the input pulse is formed. In this way voltage levels, representing bits, are converted into pulses in the delay line at the clock frequency.

Once a bit has been clocked into the line, it will reappear at the end (right hand flip-flop) of the external shift register exactly 1024 clock pulses later. By counting the clock pulses after the one in which the pulse was entered, the bit may be identified. To address the entire contents of the memory system a pulse-counting scheme is used, as shown in Figure 11. The clock pulses are counted by a combination of flip-flop elements called, collectively, counters. There are two counters, a Bit Counter and a Word Counter, as shown. Together these counters make up the Address section of the block. To design a counter the sequence of consecutive output states for all elements is specified, and the logical expression for the connection of the elements is derived by means of a transition map, a Karnough map technique modified for use in designing sequential circuitry. The Bit Counter will be so designed, and the design of the Word Counter is derived by logical induction. The Bit Counter is to count 16 clock pulses, which represent the number of bits in each buffer word; therefore the counter must be capable of assuming 16 distinct output states. The minimum number of flip-flops which may be used are four, as there are two output states that each may assume, and $2^4 = 16$ possible combinations of the four. The counter is to count in binary form, as shown in Table 3. The flip-flops are labeled A, B, C and D respectively, and the state of the count may be determined by examining the output

states of the four flip-flops. From Table 3 the transition maps of Figure 12 are derived. The input gating to the various flip-flops are determined as follows:

D: Set D = every input pulse

Reset D = every input pulse

or, trigger D on every input pulse.

C: Set C = D

Reset C = D

or, trigger C when D is logical one. When D is logical one, it will be triggered to zero on the next clock pulse alpha transition; therefore $\bar{D}$ will go through an alpha transition, and this transition, termed αD will be used as the transition input to C.

B: Set B = CD

Reset B = CD

or, trigger B = CD. When CD = 1, the next clock pulse will cause D to trigger to zero, which causes C to trigger to zero. The αC transition may then be used to trigger B.

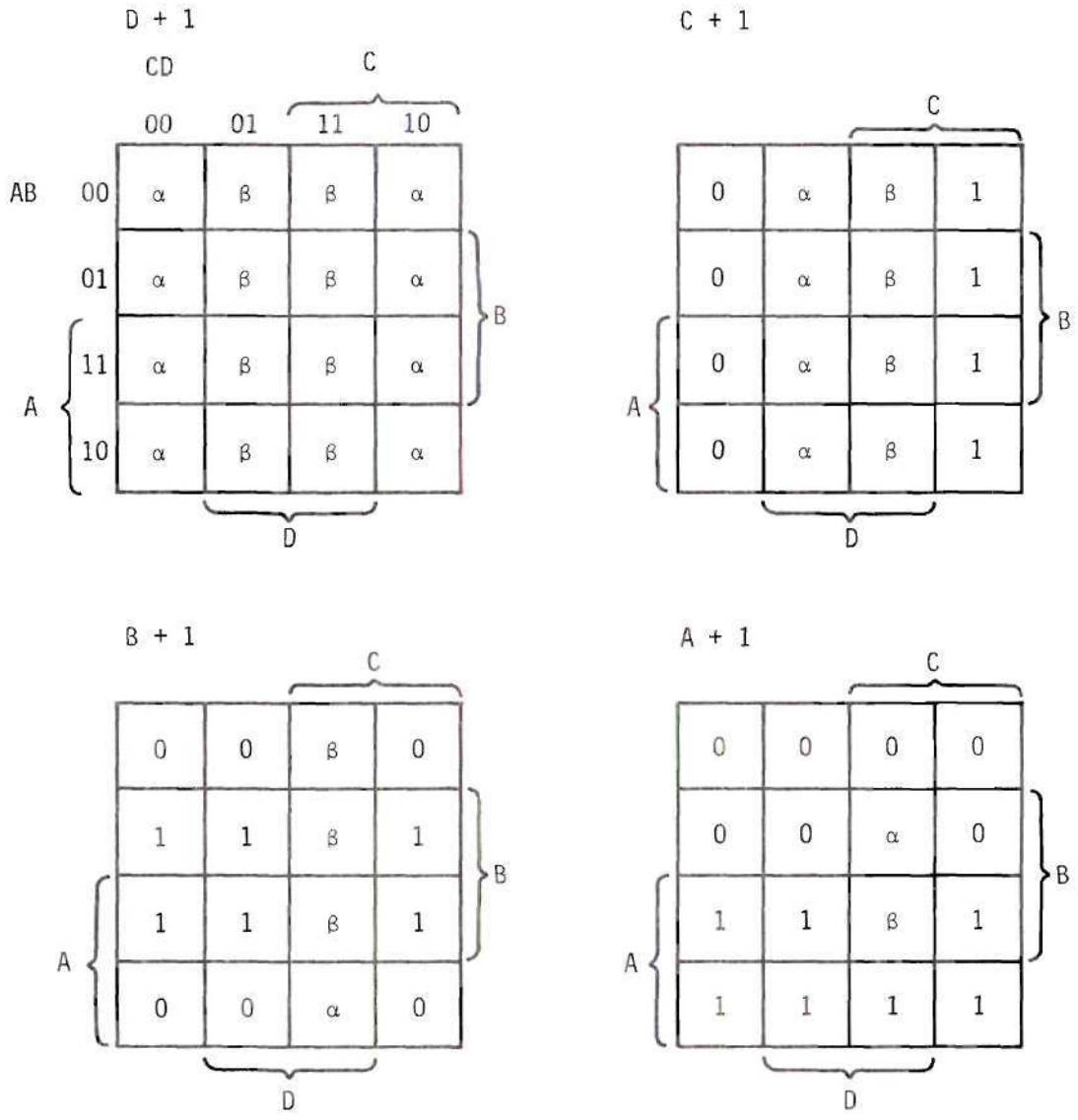
A: Set A = BCD

Reset A = BCD

or, trigger A = BCD. Using the same reasoning as for B, αB is used to trigger A.

Table 3. Bit Counter State Sequence

Flip-flop A	B	C	D	Decimal Equivalent
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	0	0	0	



NOTE: x + 1 = STATE OF ELEMENT x AFTER ONE TRANSITION PULSE IS APPLIED.

α = CHANGE FROM LOGICAL ZERO TO LOGICAL ONE

β = CHANGE FORM LOGICAL ONE TO LOGICAL ZERO

1 = REMAIN LOGICAL ONE

0 = REMAIN LOGICAL ZERO

Figure 12. Counter Flip-flop Transition Maps.

The design may be extended to any arbitrary number of flip-flops when a power of two is to be counted. The Word Counter is to count the groups of 16 bits which constitute a buffer word. There are 64 such words to be counted; therefore six flip-flops are used in the Word Counter to give $2^6 = 64$ discrete count states. As the bit counter resets from the 1111 state to the 0000 state, $\bar{0}$ of the last flip-flop goes through an alpha transition, and these transitions are counted by the Word Counter. The Word Counter will count from 0 to 63, (000000) to (111111) and reset, beginning the count cycle again. Therefore, the Word Counter recycles every 1024 clock pulses, which is the time required for one circulation of a bit in the memory system. At any point in the count a bit present at the input of the delay line will be present when that point is again reached. Thus the contents of the memory system are addressed, each bit is assigned a bit count number, and each group of bits is assigned a word count number. By determining the states of the Bit Counter and Word Counter, the identity of the bit position at the input of the delay line is completely specified.

There are two Bit Counter states of interest in the operation of the buffer, state 0(0000) and state 11(1011). These states are detected by ANDing together the particular output states of interst so that a gate output is logical one when the counter is in that state. Two four input gates are used to detect the desired states, as shown. The output of the gates are buffered to provide driving capability and to make available both voltage level forms of the gates. The gates will be refered to as the "zero gate" and the "eleven gate"

respectively in discussions that follow. It is important to note that the zero gate output is logical one when the zero position bit is in the last position of the External Register, and that the output of the eleven gate is logical one when the eleventh bit is in the last position of the External Register.

The Memory and Address Logic Block serves to store and identify information placed in it; to get part of this information into the sytem is the function of the Input Logic Block, which is described in the next section.

Input Logic Block

The Input Logic Block is shown in Figure 13. The block consists of the Input Register, the Two's Complement circuit, the Full Adder, the Address Equal circuit and the Inhibit Load circuit. Input words from the computer are set into the Input Register, a 17-place register, which has two parts: the address section and the number section. The address section is made up of the first (from left to right) six flip-flops, which store the address of an input word. Because the address is not stored in the buffer memory system, this section is not shifted. The address is set in most significant bit at the left. The number section contains the remaining 11 flip-flops, and, because the number is to be combined with a current number and stored, this section may be shifted. The number is set in so that the sign bit is in flip-flop number seven (from the left) and the ten number bits are in the remaining flip-flops, least significant bit at the right.

The function of the Inhibit Load circuitry is to enable the

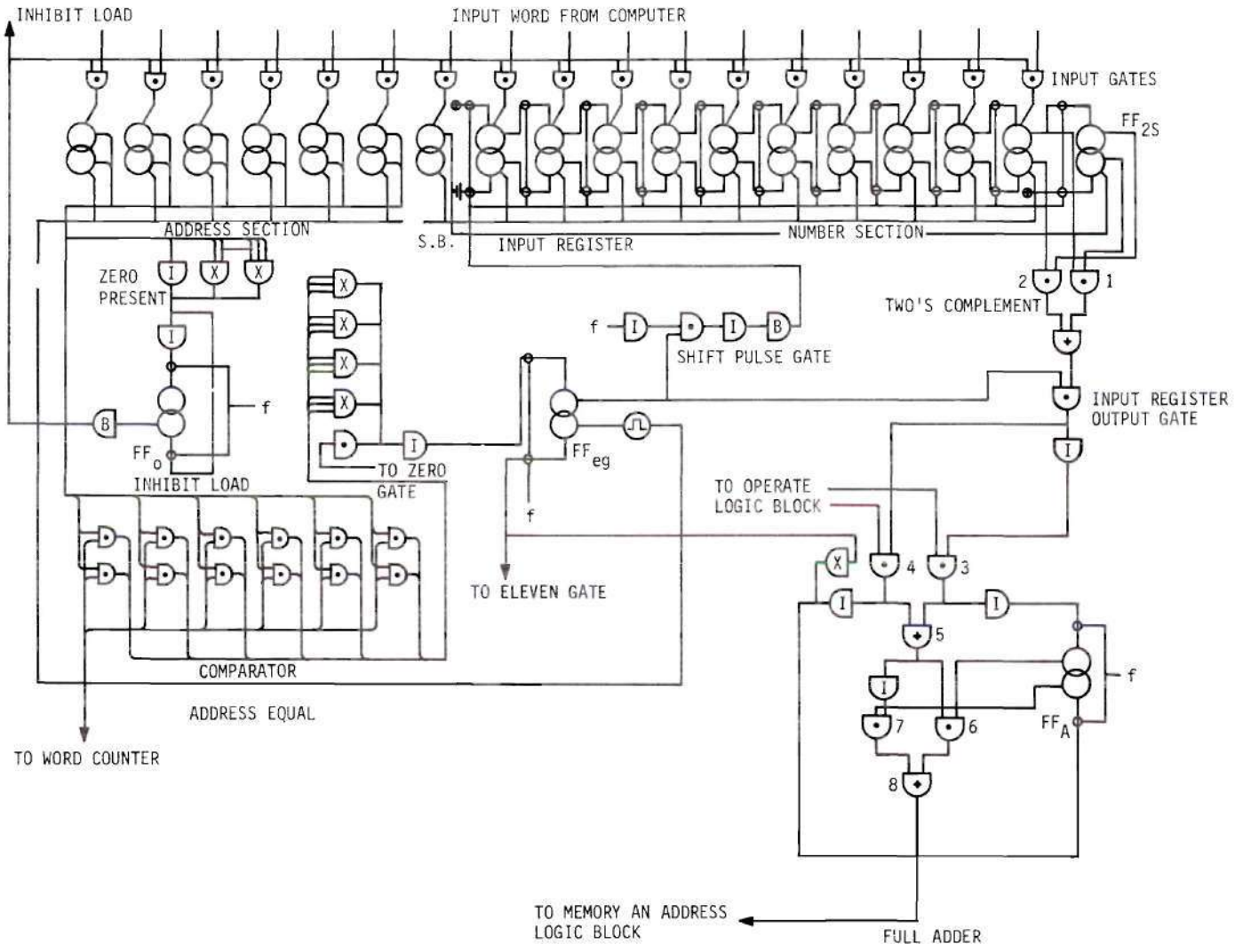


Figure 13. Input Logic Block.

loading of input words to the Input Register and to prevent loading when the Input Register contains a previous input word. The Inhibit Load circuit consists of the input gates and the zero present gate and flip-flop (FF_o). Input words are D. C. set into the Input Register flip-flops through the input gates. The Input Register is always reset before a word is loaded in, so that only the asserted form of the input word need be set in. The input gates are enabled by $\bar{O}$ of FF_o, acting through a buffer. When FF_o is reset, $\bar{O}$ is minus and the output of the buffer is plus; thus the input gates are disabled. When FF_o is set, $\bar{O}$ is plus and the buffer output is minus, enabling the gates.

FF_o is set by the zero present gate, which detects all zeros in the address section of the Input Register. As stated previously, the Input Register is reset before an input word is loaded in; more precisely, it is reset by the Address Equal circuit after each input word has been loaded in and processed into the buffer memory system. The resetting of the Input Register signifies that the previous word is gone, and another may be loaded in. When the Input Register is reset, all of the $\bar{O}$ outputs of the flip-flops are minus, thus enabling the zero present gate, which has its inputs connected to the $\bar{O}$ outputs of the address section of the register. The output of the gate is plus, which is inverted to minus to enable the set level of FF_o. FF_o then sets on the next clock pulse, and the Input gates are enabled. When any non-zero address is loaded in, the zero detect gate output goes to minus, enabling the reset level of FF_o. On the next clock pulse FF_o is reset, and the input gates are disabled, preventing the

loading in of more words unti the present word is processed and FF_o set again.

Once a word has been loaded into the Input Register, it is to be combined with the contents of the similarly addressed word space in the memory. This is detected by the Address Equal circuit, which consists of a six-place comparator and flip-flop (FFeq). When the state of the Word Counter is equal to the state of the address section of the input, the associated word space has its zero bit position in the last flip-flop of the External Line Register, thus in position for the two words to be combined. The equality of the Word Counter and the address section of the Input Register is detected by the comparator circuit. The circuit compares the states of the two numbers on a bit-by-bit basis. If all of the bits are equal, then the two numbers are equal. The comparator was designed using the Karnough map technique, as shown in Figure 14. From the map the logical expression for equality of two bits is:

$$F = AB + \bar{A}\bar{B} \tag{3}$$

Given n such pairs of variables, equality is assured when all pairs are equal, or:

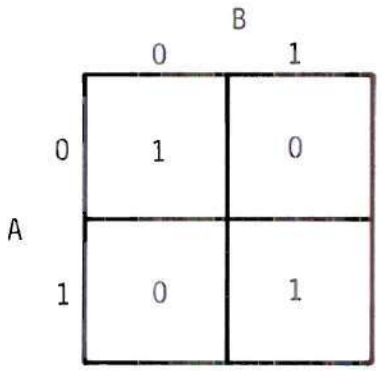
$$F \text{ equal} = (A_1B_1 + \bar{A}_1\bar{B}_1)\cdot(A_2B_2 + \bar{A}_2\bar{B}_2)\cdots \cdot\cdot(A_nB_n + \bar{A}_n\bar{B}_n) \tag{4}$$

$$\text{then } \bar{F} \text{ equal} = (\bar{A}_1 + \bar{B}_1)\cdot(A_1 + B_1) + (\bar{A}_2 + \bar{B}_2)\cdot(A_2 + B_2) + \cdots (\bar{A}_n + \bar{B}_n)\ (A_n + B_n) \tag{5}$$

$$= \bar{A}_1B_1 + A_1\bar{B}_1 + \bar{A}_2B_2 + A_2\bar{B}_2 + \cdots \bar{A}_nB_n + A_n\bar{B}_n.$$

VARIABLE STATES		OUTPUT STATE
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

TRUTH TABLE FOR A = B



KARNOUGH MAP OF F

Figure 14. Derivation for Equality of Two Variables.

For a six-place comparator circuit the expression is:

$$\bar{F}_{\text{equal}} = \bar{A}_1 B_1 + A_1 \bar{B}_1 + \bar{A}_2 B_2 + A_2 \bar{B}_2 + \bar{A}_3 B_3 + A_3 \bar{B}_3 + \bar{A}_4 B_4 + A_4 \bar{B}_4 + \bar{A}_5 B_5 + A_5 \bar{B}_5 + \bar{A}_6 B_6 + A_6 \bar{B}_6.$$

The ANDed products are formed by 12 two-input AND gates, two gates forming the expression $\bar{A}_i B_i + A_i \bar{B}_i$ for each of the six pairs of variables. The products are then ORed together in a 13-input OR gate. The thirteenth input is from the zero gate. This is done to ensure that equality will not be indicated if a word should be loaded in while the Word Counter is in the same state as that expressed in the address of the input word, but the addressed word not be in the correct position, i.e., the Bit Counter be in some state other than zero. The output of the OR gate is then inverted so that Feq is obtained.

The output of the inverter is the enabling level for the equal flip-flop, FFeq. When the output of the inverter is minus, the Word Counter and address of the input word are equal, and FFeq sets on the next clock pulse. When FFeq is set, the first bit of the buffer word number is now at the end of the External Register, and may be combined with the first bit of the number stored in the Input Register. The numbers in the External Register and Input Register are added in the Full Adder circuit, as explained in a later section. The addition is done in series; therefore the associated bits to be added must be presented simultaneously to the Adder inputs. This means that the bits in the number section of the Input Register are to be clocked out of the register in unison with those in the External Register, starting

with the first bit. Up to this time the output of the Input Register is not to be connected to the Adder.

The output of FFeq enables two AND gates when set: the shift pulse gate, which allows clock pulses to be applied to the number section of the Input Register; and the Input Register output gate, which allows the number from the Input Register to be applied to the Full Adder circuit as it is being shifted out. The addition of the two numbers is to cease after the sign bits have been added, which occurs when the Bit Counter is in state 11. The minus output of the eleven gate enables the reset level of FFeq, which is reset on the next clock pulse. When FFeq resets, $\bar{0}$ goes through an alpha transition, which is applied to a one-shot. The output of the one-shot, which is a two-microsecond, positive level pulse, is applied to the D. C. reset inputs of the Input Register, resetting it.

The shift pulse gate is enabled when FFeq is set, and gates the next eleven clock pulses to the transition inputs of the number section in the Input Register. The gate performs a logical inversion on the clock levels; therefore the clock pulses are inverted before entering the gate. The output of the gate must be buffered to provide adequate driving capability, which again will produce the unwanted inversion. Thus the output of the shift pulse gate is first inverted and then buffered, and the correct shift pulses are applied to the Input Register. Figure 15 shows the shift pulse timing.

The numbers which are to be combined in the Full Adder circuit may have either of two sign bits which indicate stepping direction or output Gray code sequence. The selection of the sign bit which

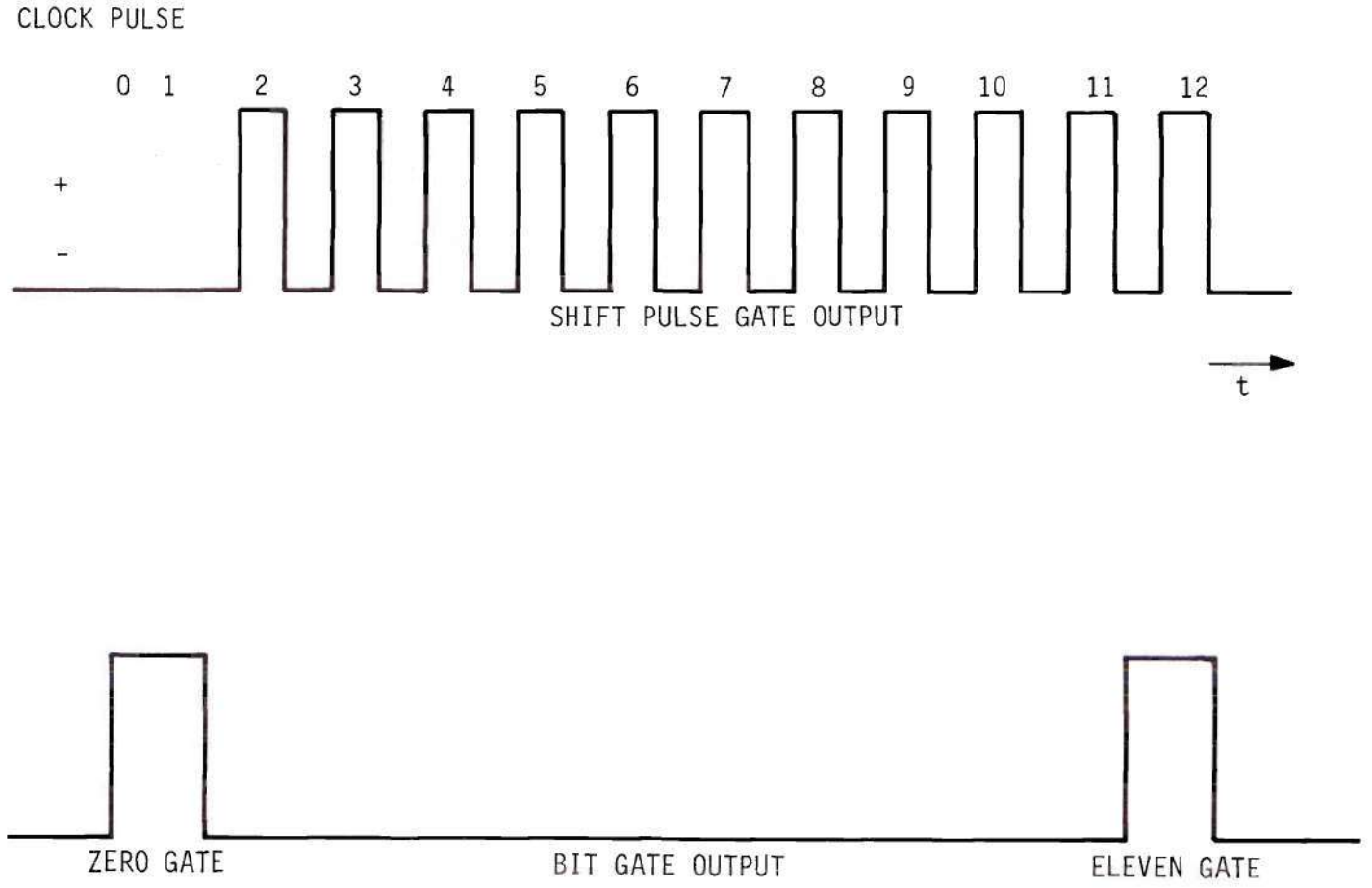


Figure 15. Input Register Shift Pulses.

designates the sequence is arbitrary; therefore a sign bit of one indicates the sequence 00, 01, 11, 10, ..., and a sign bit of zero indicates the sequence 00, 10, 11, 01, ... An input number and its corresponding buffer word space number may have the same, or opposite, sign bits. If the sign bits are alike, the two numbers are to be added; if they are unlike, the difference is to be formed. It is assumed that no number will be generated as a result of these conbinations which exceeds the maximum number (1023) which may be expressed by the ten binary number digits.

The serial subtraction of the two numbers presents certain difficulties. If it is desired to have only positive remainders as the result of a subtraction operation, the relative magnitude of the two numbers must be ascertained in order that the subtrahend be smaller than the minuend. To make such a determination, however, is very costly in terms of logic (35 AND gates); therefore another method was adopted for the numerical operation of the buffer.

A common method of subration used in digital computers is to subtracttwo numbers by first converting one of the numbers to its "complemented" form and then adding them. To define this method, the following discussion is presented.

Given a number system of radix r, which has r admissible symbols a,b,c,d,...q called digits, a number is formed in the system by forming a sequence of admissable symbols, each occupying a digit position which is numbered from some reference point. The symbols are understood to be coefficients of the radix raised to the same power as the digit position. Given two numbers in this system, A and B, where the highest order non-zero coefficient of A is in position N, and the highest order

non-zero coefficient of B is in position K, with N > K, then the remainder R = B - A may not be directly determined. However, if the magnitude of the remainder, or the difference D, is of interest, it may be found as follows.

The r's complement of any number Q in the system, which may have a maximum of m positive digit positions, is defined to be:

$$\text{r's complement of } Q = r^{m+1} - Q = C_Q \tag{6}$$

The difference between two numbers may be found if the r's complement of one is added to the other. The sum will express the difference either in r's complemented or normal form. This may be seen by considering the indicated operations using numbers A and B and assuming that the maximum number to be used has m digit positions:

$$\text{r's complement of } A = r^{m+1} - A = C_A \tag{7}$$

$$C_A + B = r^{m+1} - A + B = r^{m+1} - (A - B) = r^{m+1} - D = S$$

This is the r's complement form of the difference; therefore,

$$C_s = r^{m+1} - S = r^{m+1} - r^{m+1} + D = D \tag{8}$$

Also,

$$\text{r's complement of } B = r^{m+1} - B = C_B \tag{9}$$

$$C_B + A = r^{m+1} - B + A = r^{m+1} + (A - B) = r^{m+1} + D.$$

If the largest number to be used has m positions, then r^{m+1} is represented by a symbol in the $m+1^{th}$ position and is discarded, leaving D.

A rule may be developed for determining when to r's complement the sum by noting that if the m+1 digit position contains a zero symbol, the r's complement is to be taken. If it contains any other symbol, the first m positions express the difference in normal form, and the m+1 position is discarded. An example is presented below in the binary number system and its decimal equivalent:

Decimal r=10, m=2	Binary r=2, m=6
A = 60, B = 30	A = 0111100, B = 0011110
C_A = 100 − 60 = 040	C_A = 1000000 − 0111100 = 0000100
C_A + B = 040 + 030 = 070 = S	+B 0011110 = 0100010 = S
C_S = 100 − 070 = 030 = D	C_S = 1000000 − 0100010 = 011110 = D
C_B = 100 − 030 = 070	C_B = 1000000 − 0011110 = 0100010
C_B + A = 070 + 060 = 130	+A 0111100 = 1011110
D = 30	011110 = D

The difference, which is in r's complement form, could be left as it is and a concept of sign developed to handle the r's complemented numbers. If one of a certain group of symbols is present in the $m+1^{th}$ digit position, the number is in r's complemented form; if another one of a group of symbols is present, it is in normal form.

It should be noted that by using this convention, the sum of

r's complements is equal to the r's complement of sums, shown as follows. Where Φ represents the group of symbols specifying that the number is in r's complemented form,

$$C_A = \Phi,\ r^{m+1} - A$$

$$C_B = \Phi,\ r^{m+1} - B. \qquad (10)$$

Then, $$C_A + C_B = \Phi,\ r^{m+1} - (A + B).$$

Now, $$S = A + B$$

$$C_S = r^{m+1} - S = \Phi,\ r^{m+1} - (A + B).$$

The concept of two kinds of numbers, r's complemented and normal, is used to perform the arithmetical operation in the buffer. The sign bits which specify direction of stepping are also used to specify the form of the number associated with them. Again the choice is arbitrary; therefore, the numbers with a "one" sign bit will be r's complemented numbers, and the numbers with a "zero" sign bit will be in normal form. When this is done, all arithmetical operations regarding the combining of the buffer word numbers and the input word numbers may be reduced to the process of addition.

The input words from the computer which have a one sign bit are considered to be presented in normal form, and must be converted to r's complemented form before being shifted out to the Full Adder circuit. As the radix of the binary number system is two, the circuit which accomplishes this is known as the Two's Complement circuit. The circuit consists of two AND gates and a flip-flop, FF_{2S}, as shown.

The Two's Complement circuit was designed by noting the following. If the two's complement of a word is to be formed, this is equivalent to subtracting the word from a string of zeros:

$$\begin{array}{r} \ldots\ 000000 \\ -\ \ldots\ \underline{101100} \\ \ldots\ 010100 \end{array} \qquad (11)$$

The word to be complemented will be passed in normal (asserted) form until the first one bit occurs; this bit, is passed; all succeeding bits are passed in their opposite (negated) or "one's complemented" form; ones become zeros, and zeros become ones.

The asserted output of the last flip-flop element in the Input Register is connected to gate one, which is enabled by the $\bar{0}$ output of FF_{2S}. The negated output of the flip-flop element is connected to gate two, which is enabled by the 0 output of FF_{2S}. The outputs of the gates are then ORed and connected to the Input Register output gate. Therefore, when FF_{2S} is reset, the asserted values of any number being shifted out are passed; when it is set, the negated values are shifted out. FF_{2S} is D.C. reset by the asserted output of the sign bit flip-flop in the number section of the Input Register. As this register is reset after every word is loaded in, the asserted value of the flip-flop goes to plus, resetting FF_{2S}. Thus FF_{2S} is reset when the next word is loaded in. The set level for FF_{2S} is taken from the asserted output of the last Input Register flip-flop. The transition input for FF_{2S} is taken from the output of the shift pulse gating, so that it operates in synchronism with the bits being shifted out of the Input Register.

If the number loaded in has a one sign bit, then the two's complement of the word is to be formed as it is shifted out of the Input Register. FF_{2S} is initially reset, enabling gate one, and the asserted form of the word is passed. When a "one", or minus, is reached, the set level of FF_{2S} sets, enabling gate two and passing the negated value of the remainder of the word. If the input word is logical zero, or plus, no complementation is done, and FF_{2S} is held D.C. reset by the plus asserted output of the sign bit register.

The sign bit is not shifted out with the input number, but it is necessary that it be carried at the end of the number. This is accomplished by connecting the set and reset levels of the most significant bit flip-flop to plus and minus respectively. As the number is shifted out, the eleventh bit will have an asserted value of plus and negated value of minus. If the sign bit of the word is zero, no complementation was done, FF_{2S} is reset and the asserted or plus value is passed. If the sign bit is one, then complementation has been done (unless the number was zero), and the negated, or minus value is passed. Once out of the Two's Complement circuitry the input number and the buffer word number are to be added,and this is done in the Full Adder Circuit.

The Full Adder Circuit has two inputs, one from the Operate Logic Block, and one from the Input Register. In order that the most recent buffer word be added to the input word, the buffer word is first fed into the Operate Logic Block. The output of the Operate Logic Block then represents the most recent state of the buffer word. The output of the Full Adder is connected to the Memory and Address

Logic Block for storage.

The addition of two binary bits is summarized as follows:

1	1	0	0
+1	+0	+1	0
0, carry one	1	1	0

Any carry is to be added to the sum of the next two bits, and so on. There are three variables to be added: the two input variables, A and B, and the input carry C_i. The products formed are the sum, S, and the carry out C_o. The sum represents the addition of the input number and buffer word number and is stored. C_o is stored in the Full Adder flip-flop, FF_A, becoming C_i for the next incoming set of variables. The logical expression for the design of the Full Adder is derived by means of the Karnough maps of Figure 16 (a) and (b). From these maps,

$$S = C_i\bar{A}\bar{B} + C_iAB + \bar{C}_i\bar{A}B + \bar{C}_iA\bar{B}$$

and,

$$C_o = AB + C_iB + C_iA.$$

Using the expression for C_o, the transition map for FF_A may be drawn, as shown in Figure 16 (c), which gives:

$$\text{Set } FF_A = AB$$

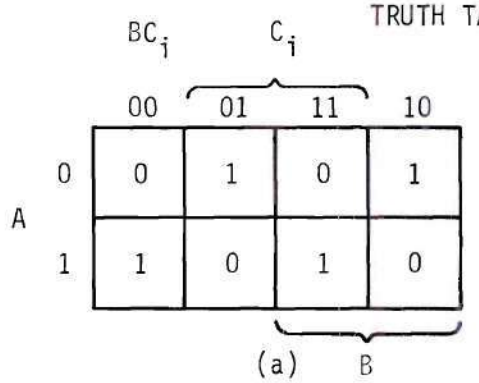
$$\text{Reset } FF_A = \bar{A}\bar{B}$$

To prevent any carry of one, which might be present in FF_A, from being added to the Gray code bits, FF_A is also reset by the eleven gate. This occurs after the last desired addition (the sign bits) and ensures

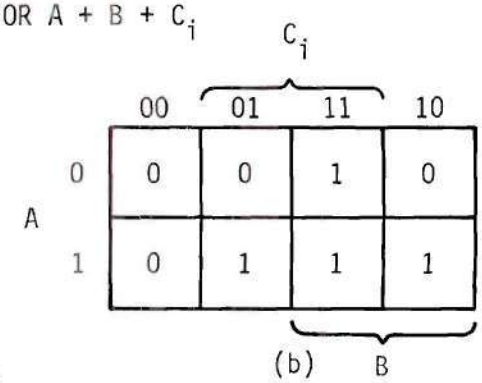
VARIABLE STATES			OUTPUT STATES	
A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

NOTE: S = SUM
C_i = CARRY IN
C_o = CARRY OUT

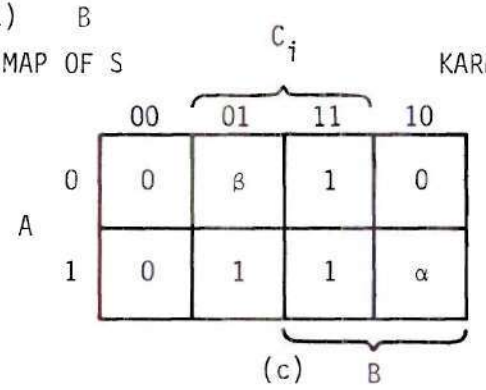
TRUTH TABLE FOR A + B + C_i



KARNOUGH MAP OF S



KARNOUGH MAP OF C_o



TRANSITION MAP FOR FF_A

Figure 16. Derivation for Full Adder Circuit.

that FF_A is reset at the beginning of the next addition. Therefore,

$$\text{Reset } FF_A = \bar{A}\bar{B} + \text{eleven gate.}$$

The Full Adder was implemented as shown, using only two input AND gates and inverters. Both the asserted and negated forms of all input variables are present by means of inverters. The implementation is derived as follows:

Gate	Input	Output
3	A, B	AB
4	$\bar{A}, \bar{B}$	$\bar{A}\bar{B}$
5	AB, $\bar{A}\bar{B}$	$AB + \bar{A}\bar{B}$
6	C_i, $AB + \bar{A}\bar{B}$	$C_iAB + \bar{A}\bar{B}C_i$
7	$\bar{C}_i$, $\bar{A}B + A\bar{B}$	$\bar{C}_i\bar{A}B + \bar{C}_iAB$
8	$\bar{C}_i\bar{A}B + \bar{C}_iA\bar{B}$, $C_iAB + \bar{A}B\bar{C}_i$	$C_i\bar{A}\bar{B} + C_iAB + \bar{C}_i\bar{A}B + \bar{C}_iA\bar{B}$

FF_A is set by the inverted output of gate three, and reset by the inverted output of gate four ORed with the eleven gate.

The operation of load-in and up-date has now been described. As soon as information is in the buffer, it must be converted to a usable form, that is, to a sequence of Gray code bits. This is the function of the Operate Logic Block, discussed in the next section.

Operate Logic Block

The function of the Operate Logic Block, shown in Figure 17, is to convert a number, carried in the number position of each buffer word, into a sequence of Gray code bits. The generation of any pair

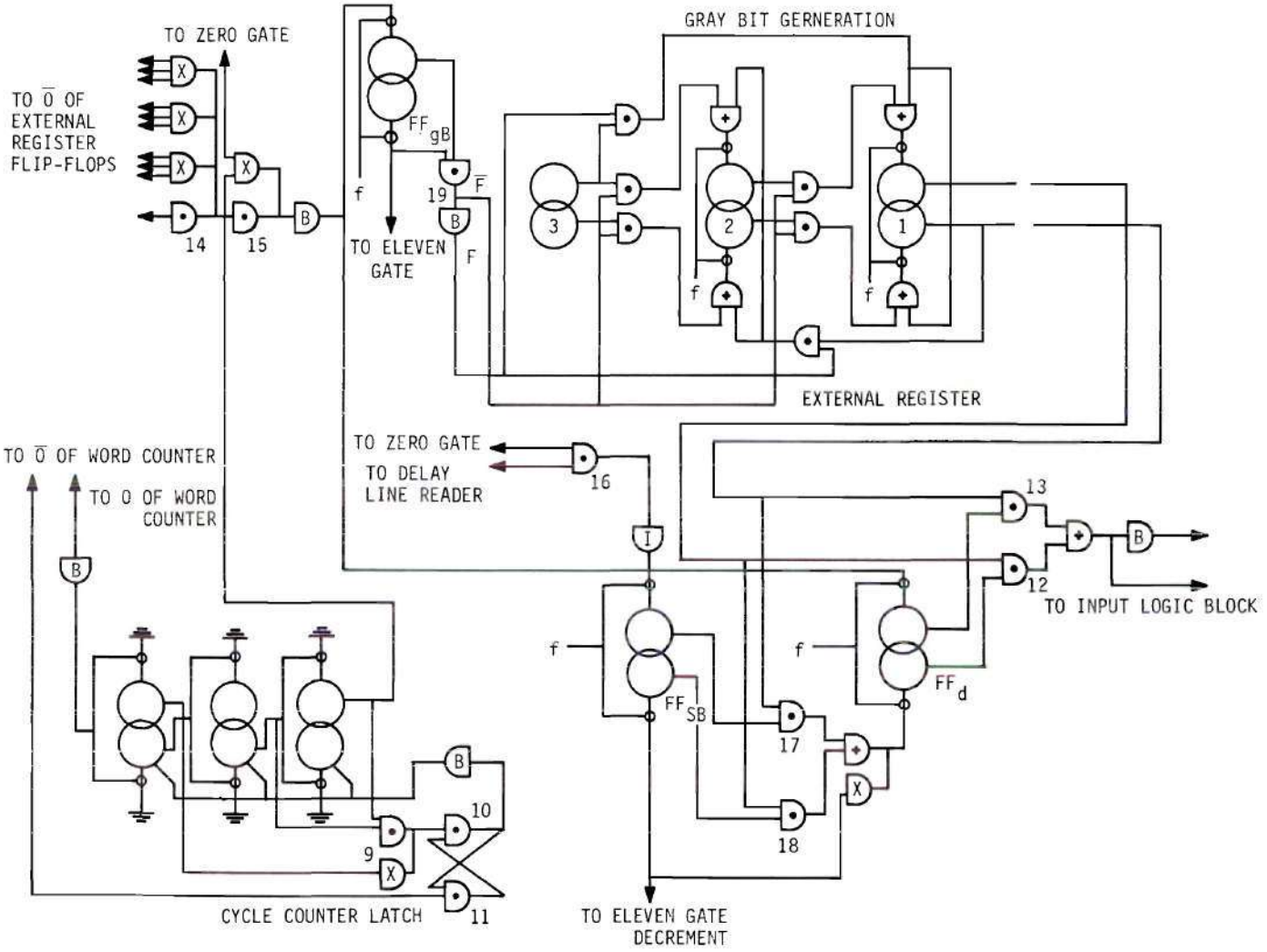


Figure 17. Operate Logic Block.

in the sequence of Gray code bits is based on the previous Gray code bits and the current sign bit of the number. The sign bit is carried in position 11 of the buffer word, as previously stated, and the Gray code bits, in positions 12 and 13. The Motor Output Logic Block then reduces the Gray code bits to voltage levels which drive the stepping motors. It is desired to step each motor every ten milliseconds; therefore the Gray code bit sequence must be generated at that rate.

The input to the Operate Logic Block is from the External Register shown in part, and the output of the block goes to the Full Adder circuit of the Input Logic Block. The Operate Logic Block consists of three circuits: the Cycle Counter, the Decrement Circuit, and the Gray Bit Generation Circuit.

New Gray bits are formed for every non-zero number buffer word every ten milliseconds, or operation cycle, and each Gray bit pair so generated represents a step taken by the motor; therefore, the buffer word numbers which express the number of steps that the associated motors are to take are reduced by one every operation cycle. In this manner the numbers are decremented to zero in one-step decrements. When a number reaches zero, no further operations are performed, and the last Gray bits generated remain constant.

The timing for the operational cycles is provided by the Cycle Counter, which counts the alpha transitions of the last flip-flop of the Word Counter in the Memory and Address Logic Block. The Word Counter resets, producing an alpha transition from the last flip-flop every 1024 clock pulses, which is every 1024/506500 seconds or

approximately every 2.02 milliseconds. Five cycles of the Word Counter are, then, 10.1 milliseconds, the desired time interval.

The Cycle Counter is a three-place binary counter which has been modified to count through five states and reset. The desired state sequences are given in Table 4. The counter counts from zero to four, and on the fifth alpha transition is reset to zero to begin the count again.

The Counter was designed as follows. When the counter is in state four, or 100, it would ordinarily go to 101 on the next alpha pulse, but it is desired to reset the counter instead. To do this the counter is allowed to count to the 101 state momentarily; this state is detected by AND gate nine; and the output of the AND gate sets a latch circuit which in turn applies, through a buffer, a plus level to the D.C. reset gates of the counter flip-flops, resetting the counter. The latch circuit is a basic storage element which behaves like a D.C. set and reset flip-flop and is made up of two AND gates as shown. It is assumed that both inputs to the latch are minus and that the output of one gate is plus, making the other minus. The latch will hold these levels until one of the inputs goes to plus. It is assumed that the input to the gate which now has a plus output is raised to plus, while the input to the other gate remains minus. The output of the gate with the plus input must now go to minus, making the output of the other gate plus. The original plus input may now be removed, and the latch will retain the new levels.

In this application latch gate 11 is constantly set at minus, from zero to four, by the plus portions of the input pulses. Latch

Table 4. Cycle Counter State Sequence

FLIP-FLOP A	B	C	DECIMAL EQUIVALENT
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
(1	0	1)	5
0	0	0	0

Note: The parenthesis indicates the momentary existence of state 101.

gate ten is then at plus, making the output of the buffer minus. When the counter goes to 101, an alpha transition has just occurred, and the input to gate 11 of the latch is minus. AND gate nine detects 101 and the output goes to plus, making gate ten minus. This output is inverted to plus by the buffer, resetting the counter to zero. When this happens the output of gate nine goes to minus, but the latch remains in the new state. The next plus level of the Word Counter output then resets the latch to its original state. When the state of the counter is 100, which occurs every fifth cycle, the asserted value of the last flip-flop is minus and remains at this level for a complete cycle of the Word Counter. This level is used to enable various gates in the remainder of the Operate Logic Block and thus serves as the timing reference.

During every operation cycle all non-zero words are to be decremented by one. This amounts to subtracting a one from all numbers with a zero sign bit and adding a one to all numbers with a one sign bit. To do this the Decrement Circuit was designed.

The operation of the Decrement Circuit may be understood by observing the following facts:

1. When a one is subtracted from a number, the bits of that number are passed in One's Complement form until the first one bit occurs. This is complemented, and the rest of the number is passed in normal form, as, for example:

$$\begin{array}{r} 110100000 \\ - \quad\quad 1 \\ \hline 110011111 \end{array} \qquad\qquad \begin{array}{r} 11010010 \\ - \quad\quad 1 \\ \hline 11010001 \end{array}$$

2. When a one is added to a number, the bits of that number are passed in One's Complement form until the first zero occurs. This is complemented, and the rest of the number is passed in normal form, as, for example:

$$\begin{array}{r} 100100111 \\ +\quad\quad\quad 1 \\ \hline 100101000 \end{array} \qquad\qquad \begin{array}{r} 10010101 \\ +\quad\quad\quad 1 \\ \hline 10010110 \end{array}$$

Both operations involve one's complementing the input bits and differ only on which bit, one or zero, the complementing ceases. These operations are accomplished by the Decrement Circuit.

The output of the External Register goes to two AND gates, numbers 12 and 13. These gates are enabled by the output of the decrement flip-flop, FF_d. When FF_d is set, gate 13 is enabled, passing the negated, or one's complemented form, of the input. If FF_d is reset, gate 12 is enabled, and the normal form of the input is passed. The outputs of the two gates are ORed together and go to the Full Adder circuit in the Input Logic Block.

If the Cycle Counter is in state four (100) and the number to be decremented is not zero, decrement operations are to be performed, commencing with the first bit of each buffer word number. These conditions are detected by AND gates 14 and 15. The inputs to AND gate 14 are from the negated outputs of the External Register flip-flops; the output is connected to one of the inputs of gate 15. The output of gate 14 is logical one, or plus, whenever the External Register contains all zeros, and is logical zero, or minus, if there is at least one non-zero flip-flop. The other two inputs to gate 15 are the zero gate

and the Cycle Counter enable level. If the Cycle Counter is in state four, the contents of the External Register are not zero and the bit count is at zero, then gate 15 will be enabled. The output of gate 15 is inverted and applied to the set levels of FF_d and the Gray bit flip-flop, FF_{GB}. If the gate is enabled, these two flip-flops will be set on the next clock pulse.

FF_d is originally reset. It is then set as the first bit of any number to be decremented is at the output of the External Register. Gate 13 is enabled, and will pass the One's Complemented form of the input word until FF_d is reset. FF_d will be reset by the first one of zero in the input word, depending upon the decrement operation desired.

The decrement operation depends upon the sign bit of the number to be decremented: minus one if the sign bit is zero (normal form), and plus one if the sign bit is one (two's complemented form). By means of AND gate 16 the sign bit is set into the sign bit flip-flop, FF_{SB}, at the same time that FF_d is set. The inputs to AND gate 16 are from the delay line reader and the zero gate. The output of the reader will be the sign bit, when the zero gate enabling level is present; therefore, the output of gate 16 is the sign bit. The gate output level is inverted and applied to the set level of FF_{SB}. FF_{SB} is set only if the sign bit is one. It is initially reset at the beginning of each operation, and thus always begins at a sign bit of zero.

FF_d is reset by either AND gate 17 or 18. Gate 17 is enabled when FF_{SB} is set, and gates the negated value of the input word. Gate 18 is enabled when FF_{SB} is reset, and gates the asserted value of the input word. The two outputs are then ORed and applied to the reset

level of FF_d. If the sign bit is one, FF_d is to be reset by the first zero of the input word, FF_{SB} is set, gate 17 is enabled, and FF_d is reset on the clock pulse following the first zero bit. If the sign bit is zero, FF_d is to be reset by the first one of the input word, FF_{SB} is reset, gate 18 is enabled and FF_d is reset on the clock pulse following the first one bit. When the end of the number is reached, the decrement operation is to cease; therefore the reset level of FF_{SB} is enabled by the eleven gate, which is also ORed with the reset levels of gates 17 and 18 for FF_d. The two flip-flops are then reset on the next clock pulse.

The new Gray code bits are also formed during the operation cycle. This is accomplished by the Gray Code Generation circuit, reproduced, in part, in Figure 17. The circuit will form the new Gray code bit sequence, based on the preceding Gray code bits and the sign bit.

When the bit counter reaches state eleven, the sign bit is in the output flip-flop of the External Register, and the Gray code bits, A and B, are in the preceding two flip-flops (labeled 1, 2, 3, respectively). If new Gray code bits are to be formed, FF_{GB} will be set, as noted previously. The asserted output of FF_{GB} is ANDed with the eleven gate by AND gate 19, enabling the Gray Code Generation Circuit as shown. On the next clock pulse the old Gray code bits will be changed to the next sequence.

The circuit was designed as follows. All possible sequences of variables of interest, F, the inverted output of gate 19, S the sign bit, A the first Gray bit, and B the second Gray bit, are shown in Table 5,

Table 5. States of Gray Code Generation Circuit Variables

VARIABLE STATES				VARIABLE STATES PLUS ONE CLOCK PULSE			
F	B	A	S	F	B	A	S
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	1	0	0
1	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0
1	0	1	1	1	1	1	1
1	1	0	0	1	1	1	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	0
1	1	1	1	1	1	0	1

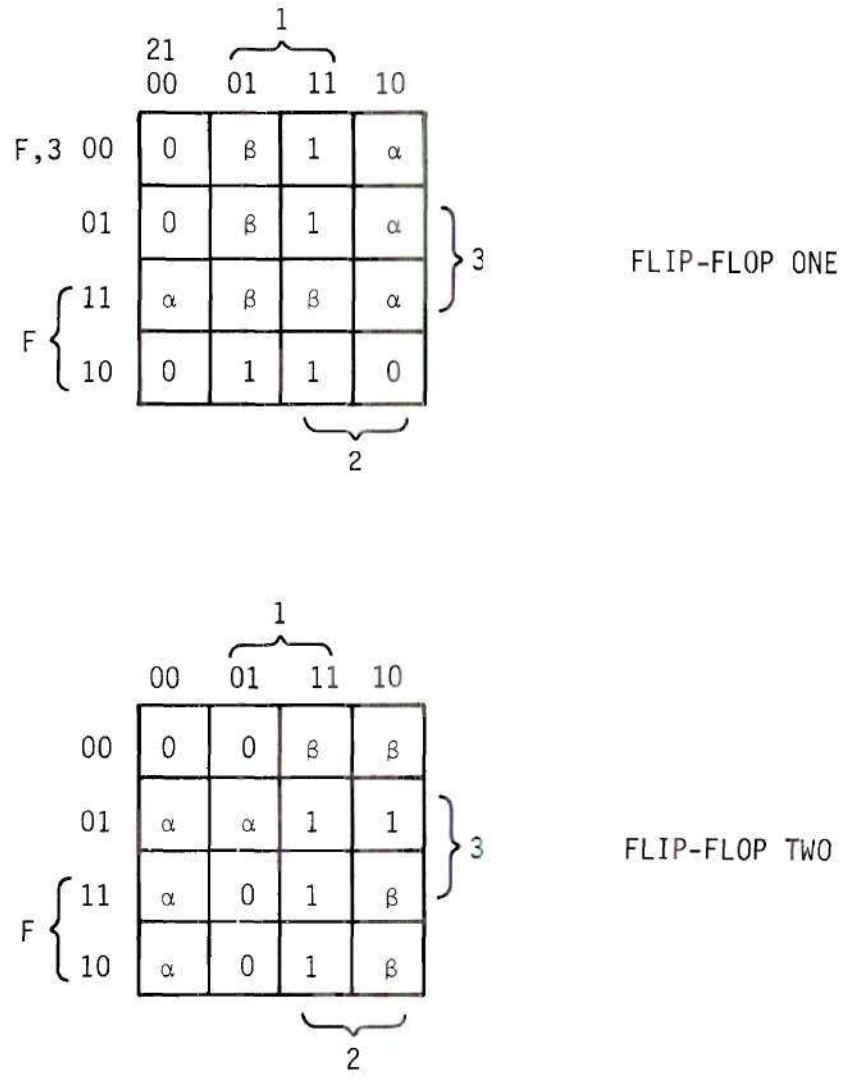


Figure 18. Gray Code Generator Transition Maps.

both before and after the twelfth clock pulse. From this table the transition maps of Figure 18 are drawn for the two flip-flops of interest.

From these maps,

$$\text{Set } 1 = 2\bar{F} + 3F$$

$$\text{Reset } 1 = \bar{2}\bar{F} + 3F$$

$$\text{Set } 2 = 3\bar{F} + \bar{1}F$$

$$\text{Reset } 2 = \bar{3}\bar{F} + \bar{1}F$$

From these expressions the circuit is implemented as shown. FF_{GB} is reset by the eleven gate level at the end of each number. The new Gray code bits have now been formed; to convert them to voltage levels applied to the field coils of the proper stepping motor is the function of the Motor Output Logic Block, as presented in the section below.

Motor Output Logic Block

The Motor Output Logic Block is shown in Figure 19. Because only one stepping motor was connected, the block consists of the output logic for one motor, which would be duplicated for the desired number of motors. Each output circuit is made up of an Address gate, four latch gates, two latches, four latch output buffers, and the Motor Driver.

The outputs of the latches are buffered and applied to the Motor Driver circuit. The Motor Driver consists of Four NPN power transistors, each of which has one field coil of the stepping motor connected to the collector in complementary pairs, as shown. Diodes are connected across

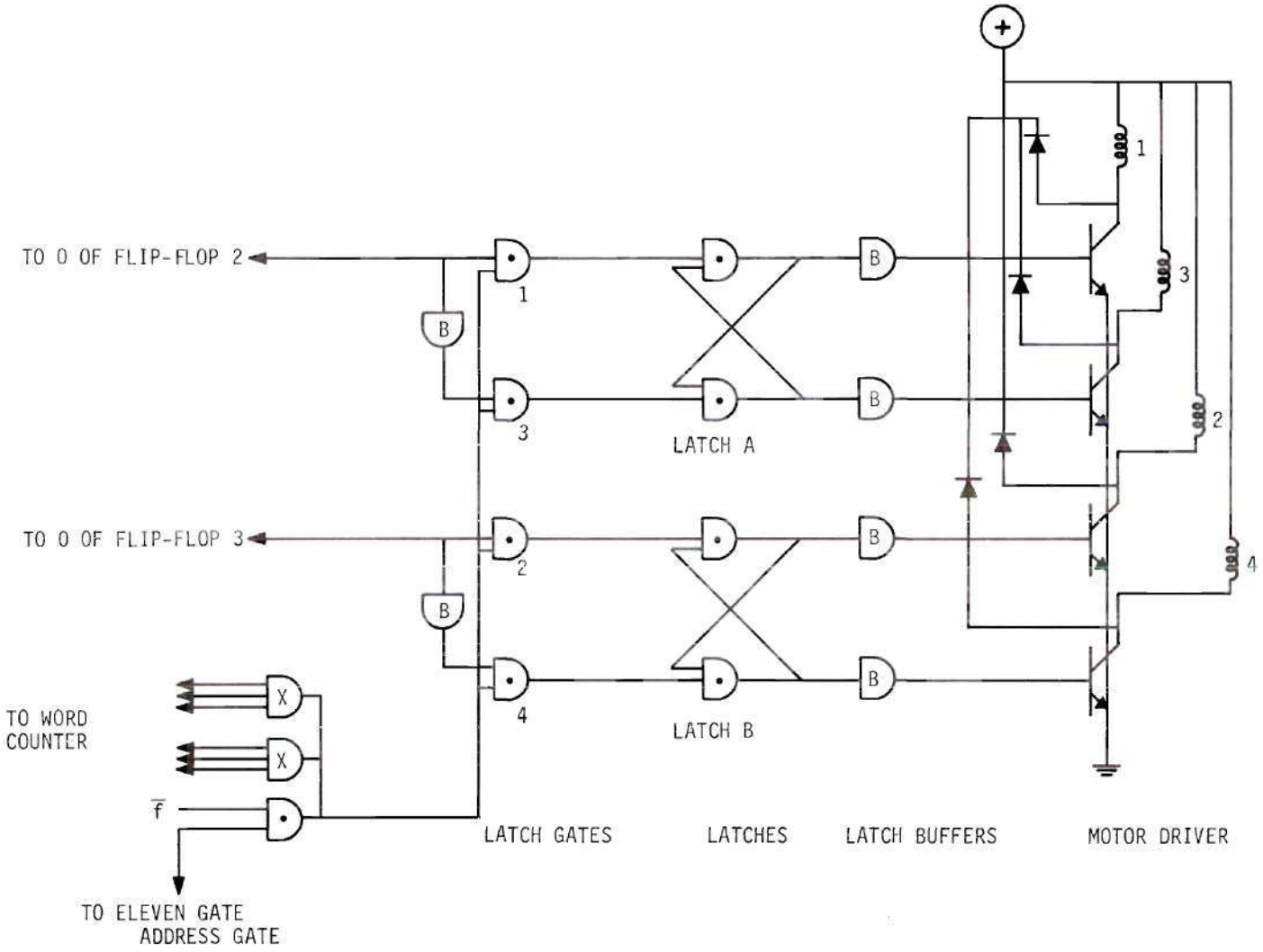


Figure 19. Motor Output Logic Block.

the field coils, so that any inductive voltage surges which result from the coil switching may be damped.

Each latch circuit stores, in asserted and negated form, one of the Gray code bits which are generated in the Operate Logic Block. Latch A stores the A bit, and Latch B stores the B bit. The outputs of the latches switch the Motor Driver transistors. The two Gray code bits are thus converted into voltage levels which drive the stepping motor.

The Gray bits are gated into the latch circuits, through the latch gates, when these gates are enabled by the Address gate. Each motor is numbered, from one to 63, to correspond to a similar address in the buffer word counter. The Address gate detects when the particular address of interest is displayed by the Word Counter in the Memory and Address Logic Block, and when the Gray bits are in the desired position in the External Register. The Gray bits will be in flip-flops two and three of the External Register when the Bit Counter is at eleven; therefore, the output level of the eleven gate is ANDed with the desired Word Counter levels in the Address gate. To prevent ambiguities, an $\bar{f}$ clock pulse is also ANDed in the Address gate. The inputs to latch gates one and three are connected to the asserted and negated outputs, respectively, of flip-flop two, and the inputs to latch gates two and four are connected to the asserted and negated outputs, respectively, of flip-flop three. The Address gate will be enabled on every cycle, thus gating the current Gray bits into the latches. Every fifth cycle the Gray bit sequence will change, stepping the motor one step.

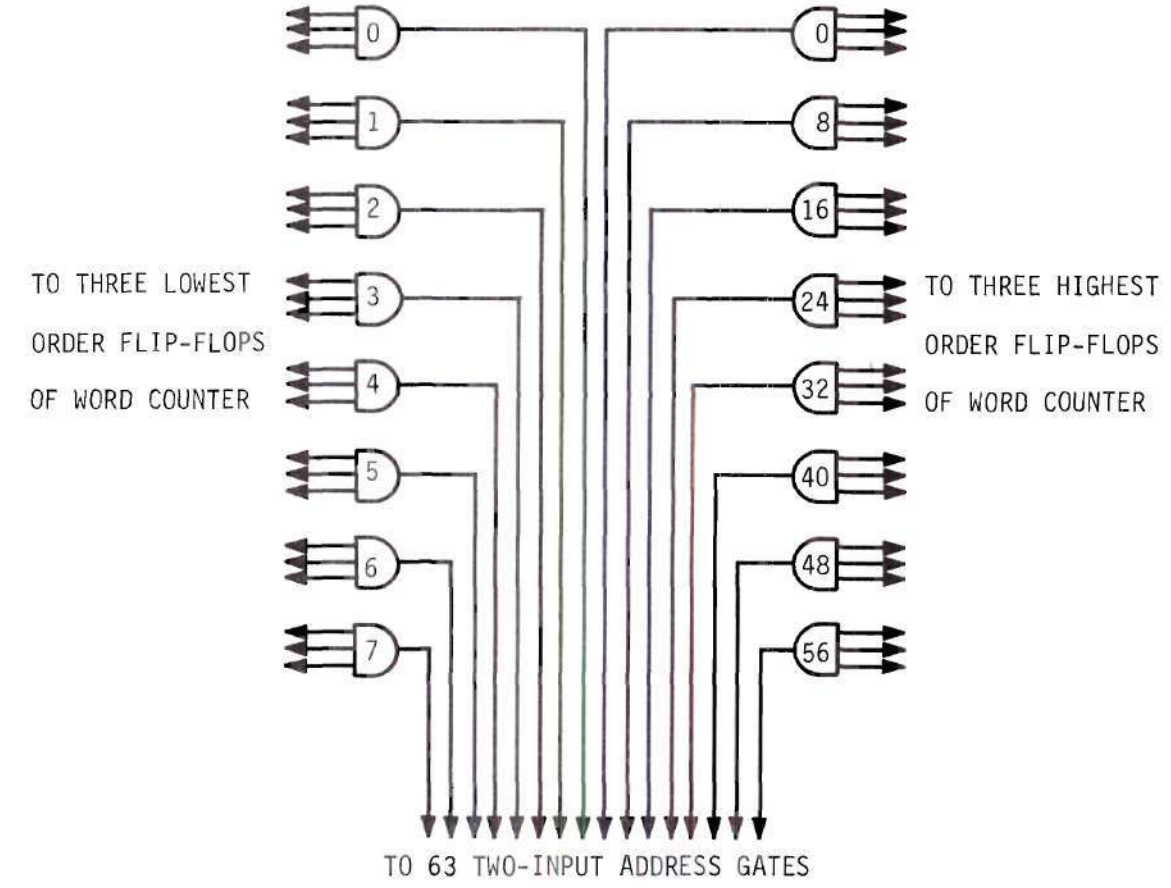
The motor drive logic may be contained in the buffer, or located remotely with the stepping motors. If located remotely, provisions must be made to ensure pulses are not degraded in passing from the buffer to the remote location. As used here, all output logic is considered to be retained in the buffer, with the collector leads of the Motor Driver transistors furnished to the remote stepping motors.

The addressing scheme used here for a single motor and requiring a six-input gate for each address would not be the most efficient for a large group. To address all 63 motors, the Word Counter may be divided into two halves and each half addressed separately. Figure 20 shows such an address matrix, which is termed an octal address system.

Buffer Construction

The buffer, whose design has been explained in previous sections, was constructed using a commercially available brand of card-mounted, integrated-circuit logic, samples of which are shown in Figure 21. The cards were plugged into connectors, and the latter mounted in an aluminum chassis. Wiring was done using the wire-wrap technique, which afforded dense, yet reliable, connector interconnections. Special circuits, such as the delay line input-output circuits, were made up on general-purpose plug-in cards. The dealy line was mounted in the rear of the chassis.

The assembled chassis was mounted in a cabinet equiped with front and rear panels, as shown in Figure 22. The front panel contains the two operating controls, an on-off switch and a reset pushbutton. The on-off switch controls the application of power to the buffer; when pushed, the pushbutton will hold the last flip-flop of the Input



NOTE: NUMBERS IN GATES
REFER TO WORD COUNTER
STATE DETECTED.

Figure 20. Octal Address Matrix.

Register D.C. reset, thus emptying the entire contents of the delay line. The philosophy here was that the less external control presented, the smaller the chance of unwanted operator interference.

Inputs from the control computer are brought in by means of a 24-pin connector mounted in the rear panel; outputs to the motors are fed out by means of five-pin connectors mounted in the rear panel (one shown). D.C. power for the buffer may be furnished from external sources or by self-contained power supplies. Space was left on the chassis for such power supplies, and an opening for an A.C. power cord provided in the rear panel.

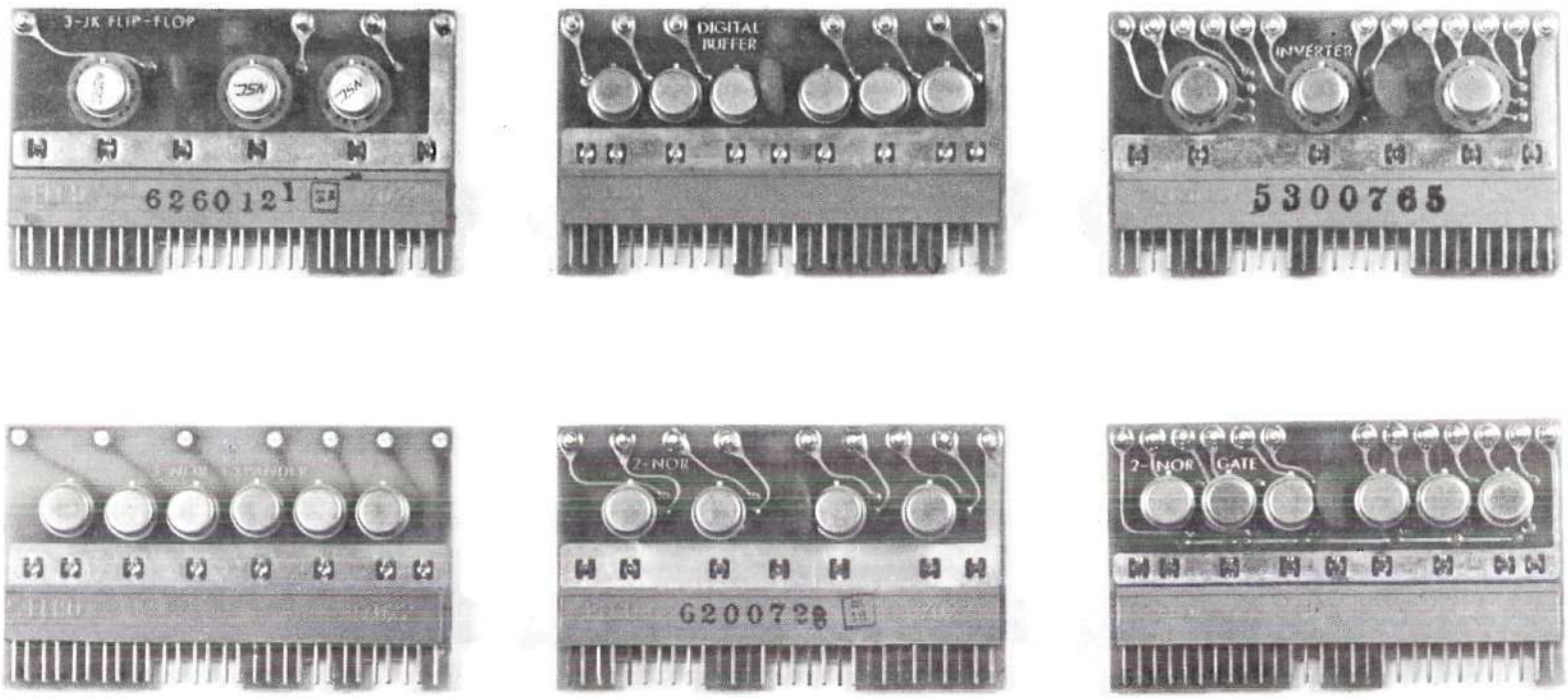
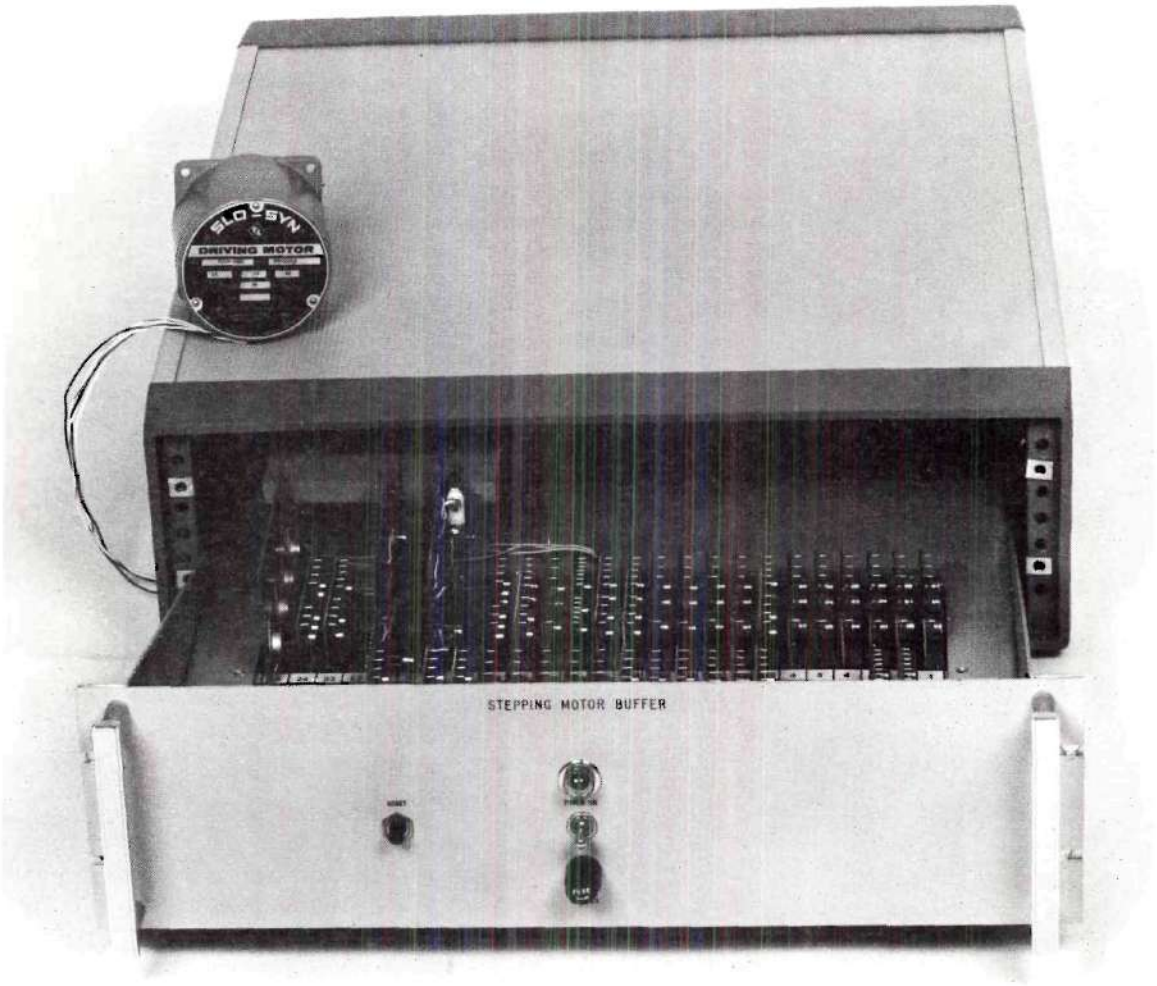
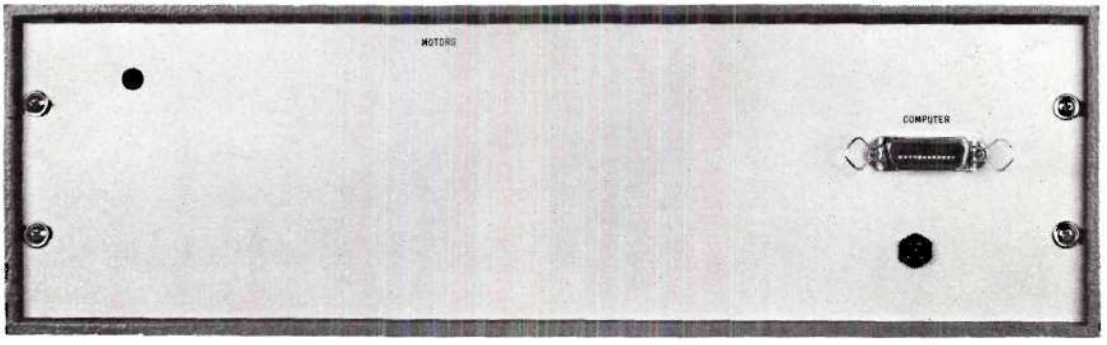


Figure 21. Logic Cards.



(a)



(b)

Figure 22. Stepping Motor Buffer.

CHAPTER III

EXPERIMENTAL TESTS AND RESULTS

Test Configuration

The buffer circuitry was originally placed in a five-inch deep chassis, which was inverted so that the connector pins could be easily wired. Each circuit of the various logic blocks described in the previous chapter were wired and tested on a circuit-by-circuit basis. The buffer required three power supplies: P_B, which supplied power to the integrated-circuit logic cards (3.2 volts, 1.7 amperes); P_M, which supplied power to the stepping motor (6 volts, .6 amperes); and P_D, which supplied power to the delay line driver and reader (12 volts, 100 milli-amperes).

The stepping motor was connected to the Motor Driver Circuit, and a large protractor placed on the shaft end so that degrees of rotation could be ascertained. Each step of the motor used was a rotation of 1.8 degree ± .09 degrees, or 200 steps per revolution. The steps taken for any test were then determined by noting the starting and ending points of the shaft pointer. For test purposes the stepping motor was assigned address number 26. The configuration is shown, in block diagram form, in Figure 23.

To simulate the loading in of a computer word, 17 single-pole, double-throw toggle switches were connected to the input gates of the Input Register. The switches could be set to plus or minus, thus forming

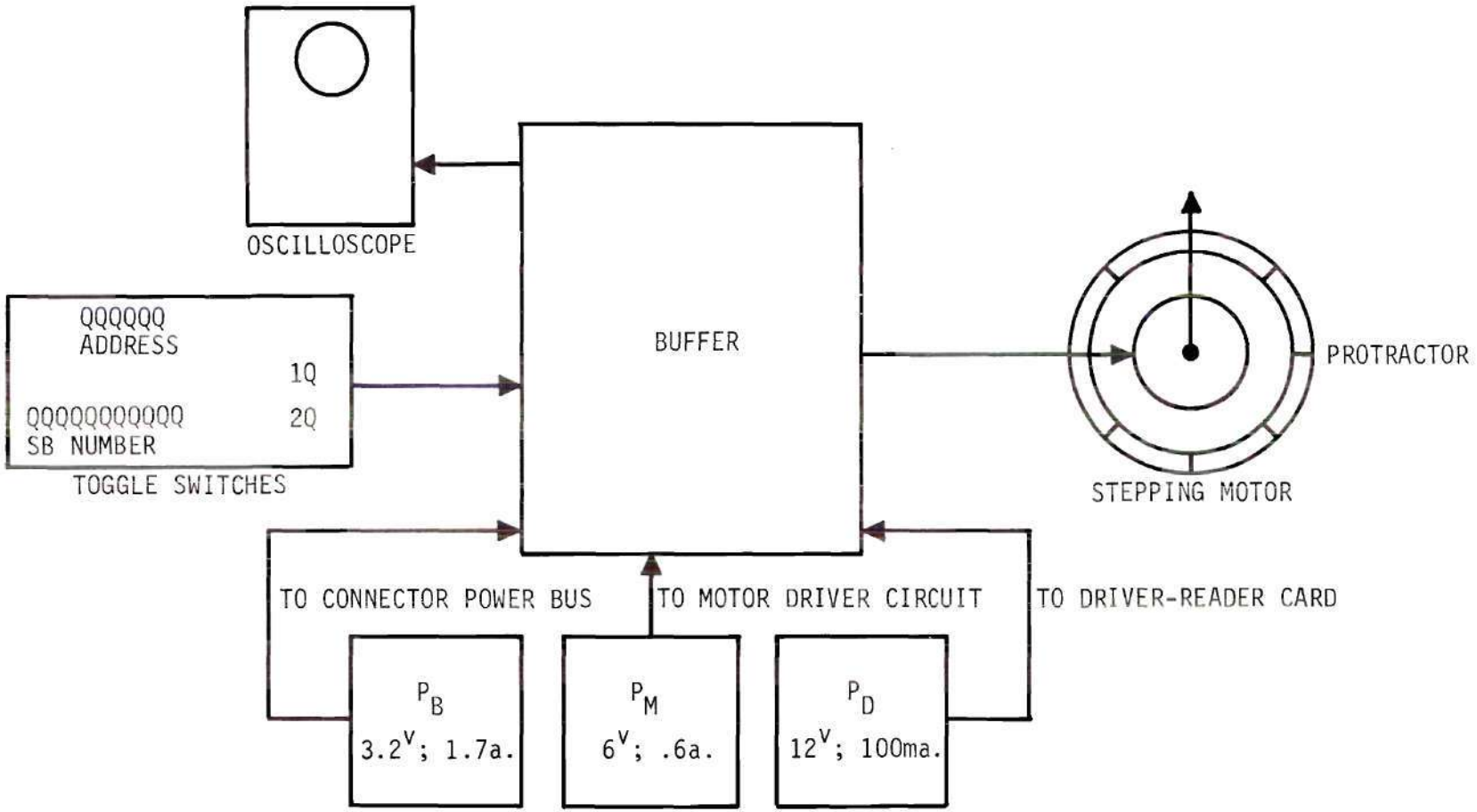


Figure 23. Test Configuration.

any desired input.

The load-in circuitry was modified as shown in Figure 24. Toggle switch one was connected to a spare input of the zero-present gate of the Input Logic Block. This switch is used to enable or disable that gate. The reset connection to the address section of the Input Register was broken and connected to toggle switch two. Toggle switch one and two were later replaced with a two position rotary switch.

To load in a word, the following procedure was followed:

1. Set toggle switch one to plus (disabling the zero-present gate; resetting FF_o, thus disabling the input gates).
2. Set toggle switch two to plus (resetting any previous contents of the address section).
3. Set toggle switch two to minus.
4. Set in the desired input word on the 17 toggle switches.
5. Set toggle switch one to minus.

The word set in on the toggle switches will be loaded into the input register in one clock pulse. When toggle switch one is set to minus, the zero-present gate is enabled, setting FF_o and enabling the input gates on the next clock pulse. As soon as the number is set in, the zero-present gate is disabled (for any non-zero address) and FF_o then resets on the next clock pulse.

Visual display of the buffer operation was provided by connecting an oscilloscope to the output of flip-flop one in the External Register.

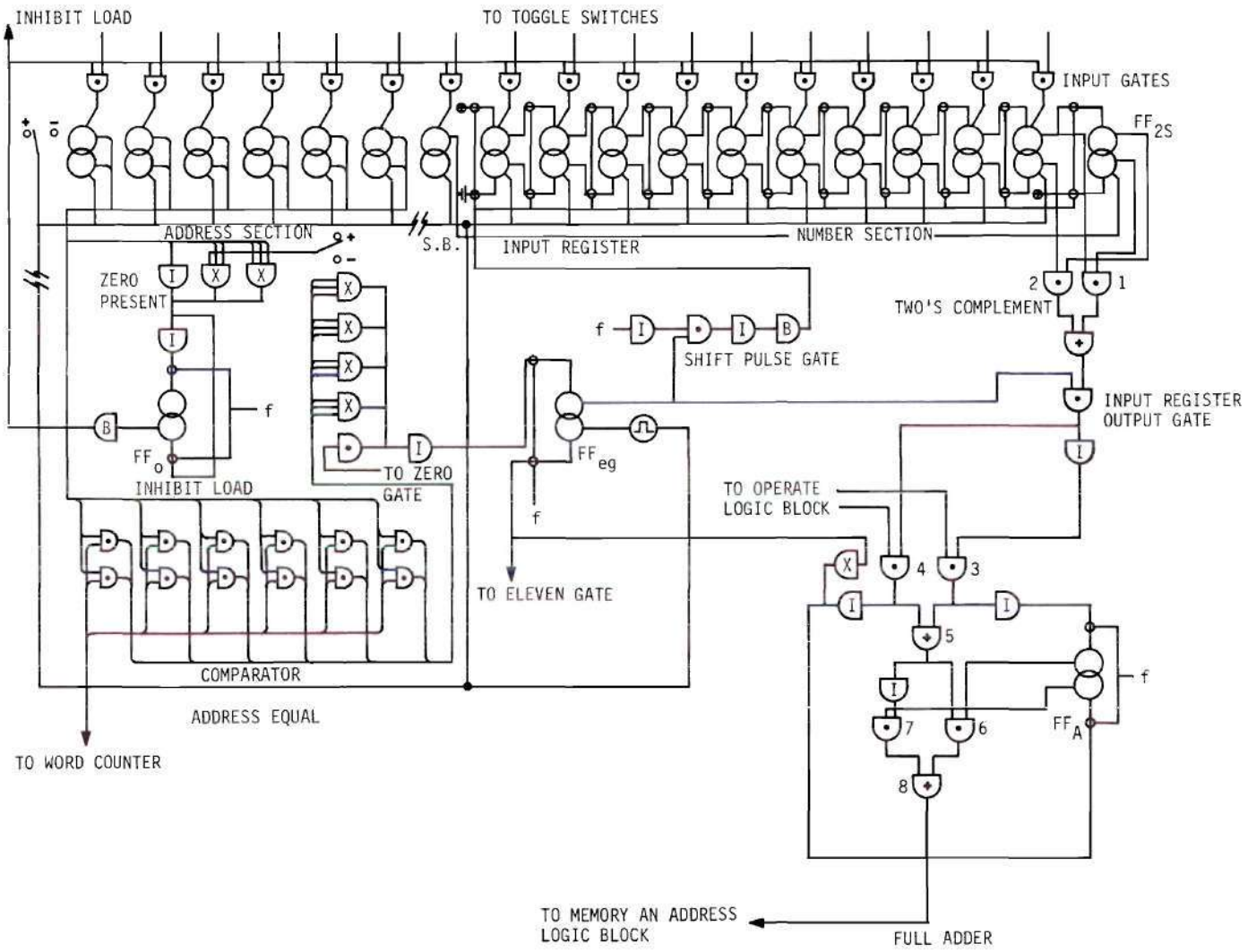


Figure 24. Input Logic Block Test Modifications.

Adjustment of the Buffer

Only one adjustment may be made, and that is to fix the length of delay of the delay line. What is of primary interest is the fact that any bit put into the delay line input reappears at flip-flop one of the External Register 1024 clock pulses later. The adjustment may be made by modifying the circuit as shown in Figure 25 and explained below:

1. Disconnect the outputs of flip-flop one in the External Register.
2. Connect the inputs of the decrement circuit to plus and minus, as shown.
3. Connect a plus voltage to the D.C. reset input of FF_d.
4. Connect one lead of a dual-trace oscilloscope to the asserted output of flip-flop one in the External Register and the other to the input of the delay line driver.
5. Set toggle switch two to plus and toggle switch one to minus. This will continuously load in any word placed on the input switches.

The delay line is then adjusted by turning the adjustment screw. The desired pattern is as shown in Figure 26, where the output of the external register leads the input to the delay line driver by half a pulse width. Several patterns of input word should be used to determine if the Delay Line Reader is operating reliably, and it may also be adjusted until the proper results are obtained.

Tests and Results

Four basic tests were performed to determine the accuracy of the

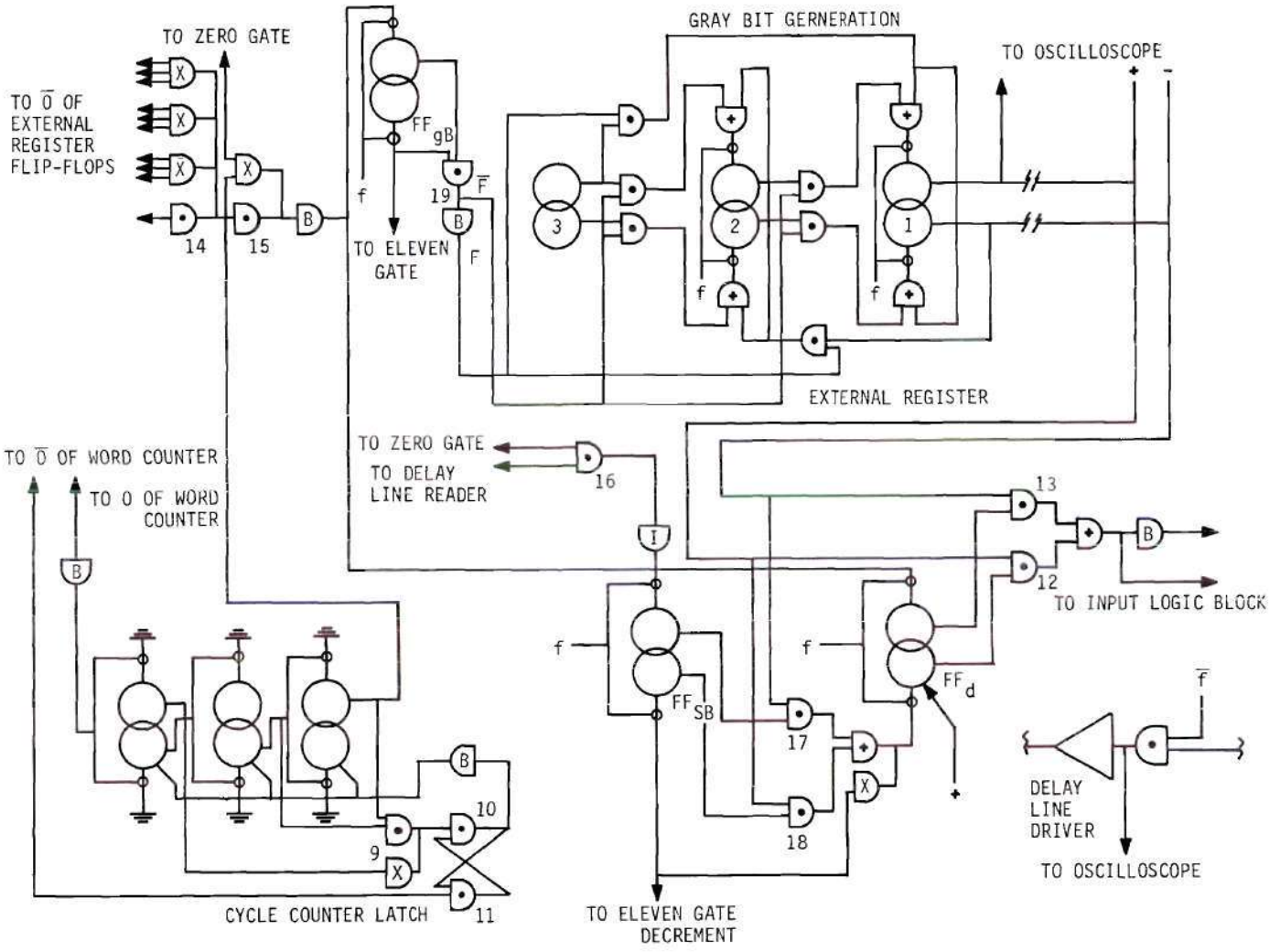


Figure 25. Delay Line Adjustment Modifications.

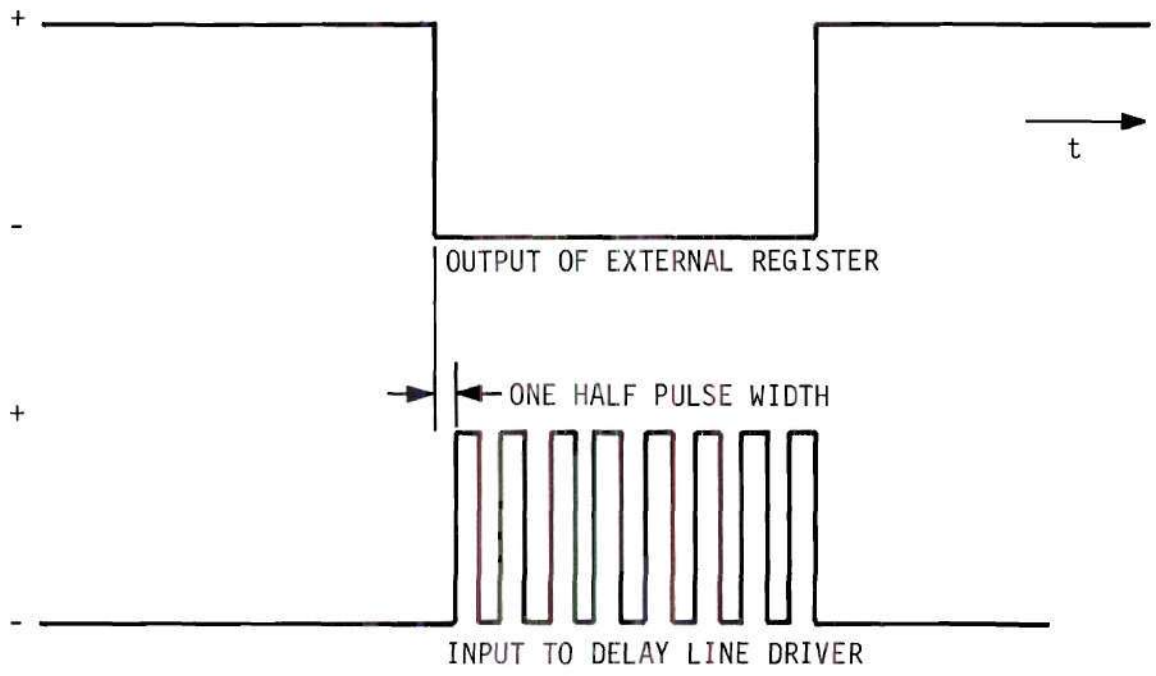


Figure 26. Typical Delay Line Adjustment Pattern.

buffer. These were:

1. Extended Accuracy Test
2. Random Accuracy Test
3. Address Accuracy Test
4. Update Test.

The Extended Accuracy Test was performed by setting the magnitude of the input word to 1000, and the address to 26. The sign bit was set alternately one and zero. The word was set in and the motor allowed to step. The sign bit was then reversed, and the motor stepped again. Each sequence of the above two numbers should result in exactly five revolutions of the stepping motor in each direction. This test was repeated 200 times, or 400,000 steps of the stepping motor, with no error.

The Random Accuracy Test consisted of setting in random numbers as input words (all with address 26) and recording the degrees of rotation of the stepping motor shaft. This was done for 39 numbers ranging in magnitude from zero to 1023, and for both sign bits. The memory was checked to see if any other address contained information. No error occurred.

The Address Accuracy Test was performed by consecutively setting in the number 1023 for all addresses from one to 63, with the exception of 26. The oscilloscope was observed to see if the input word was being decremented for each address, and the stepping motor was observed to see if any steps were taken. The number was set in with each of the sign bits for a total of 126 tests. No error was observed.

The Update Test was performed to determine if a number could be loaded into the buffer and another added (or subtracted) to it. To show that subtraction is done the magnitude of the input number was set to 1000, with one sign bit; the number was loaded in, and, as the stepping motor stepped, the same number, opposite sign bit, was loaded in. The result of the operation should result in no net steps. To show that addition is done, the number 500 was set it with one sign bit; the number was loaded in, and, as the stepping motor stepped, the same number was set in again. This was repeated for the opposite sign bit. The result should be 1000 steps or five revolutions of the stepping motor. Both tests were repeated 25 times each, and no error occurred.

Conclusion

From the results of the tests performed the design of the buffer was proved to be operable. The reliability of the buffer and its performance under adverse environmental conditions were not tested; nor was the prototype built to undergo such tests. However, based on the performance specifications of the logic used, it is believed that the present design could be readily adapted for industrial use with little modification.

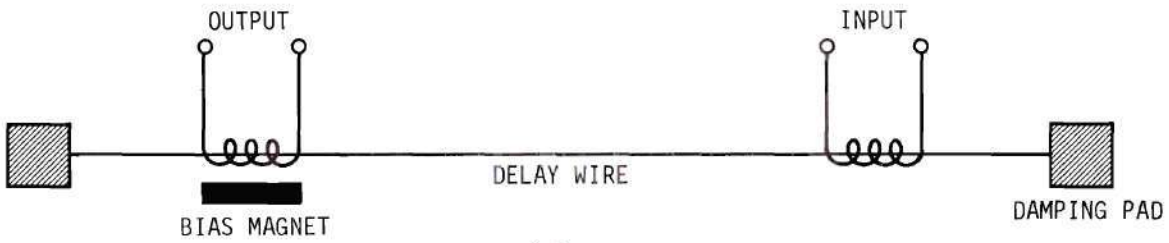
APPENDIX I

Delay Line and Associated Circuitry

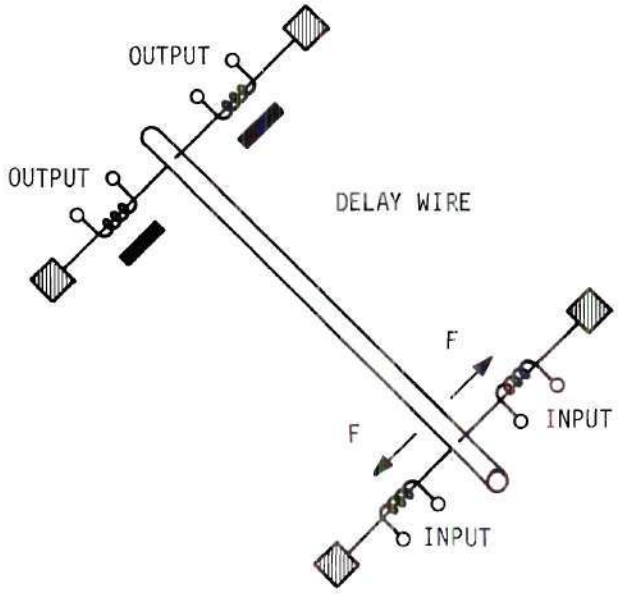
The delay line used in the construction of the buffer is properly termed a sonic magnetostrictive wire delay line. The function of the delay line is to delay the passage of electrical pulses for some time period. The number of consecutive pulses which are so delayed before the first pluse appears at the output then determines the number of pulses or information bits stored. The delayed information may be recirculated through the delay line by external circuitry, and a permanent store formed. Used in this manner, the delay line is a dynamic storage device, with information stored in time rather than in space.

Delay lines are attractive memory elements due to their low cost, reliability, simplicity and serviceability, as compared with other memory devices.[9] In per-bit cost they fall between core storage and magnetic tapes and drums.[10] However, when low capacity storage is needed, they offer perhaps the best solution.

A basic magnetostrictive delay line may be built as shown in Figure 27(a). Electrical current impulses are converted to sound waves by means of the input transducer, which consists of a coil of wire wound around the delay wire. This conversion makes use of the principle of magnetostriction: the deformation of certain materials when subjected to an electromagnetic field. The delay line is made of such a material, usually a nickel alloy. A pulse of current, impressed on



(a)



(b)

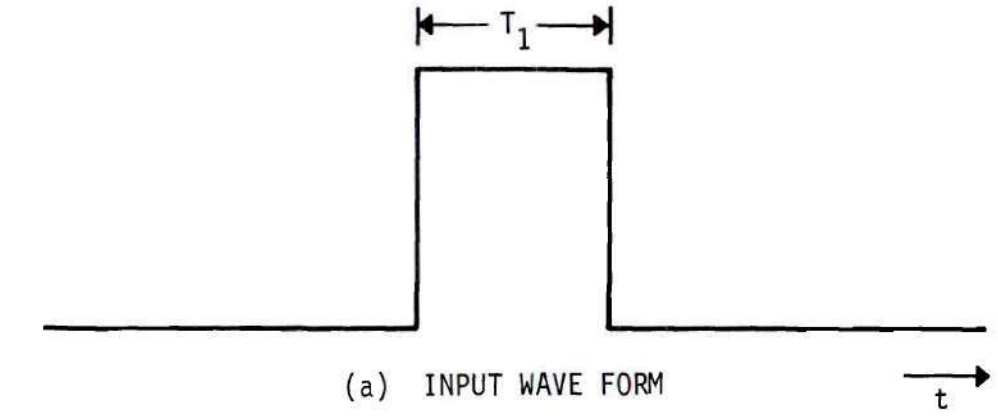
Figure 27. Delay Line Construction.

the input transducer, produces a magnetic field, which in turn produces a longitudinal strain wave in the wire. This wire travels away from the transducer in both directions, but the end of the wire is embedded in a damping material to absorb the unwanted wave. The remaining wave travels down the line to the output transducer. The output transducer is similar to the input transducer, except that a small permanent magnet located in close proximity to the coil provides a bias flux in the wire. When the strain wave arrives, the wire in the coil is deformed, changing its permeability and thus the flux linking the coil. This generates a voltage, proportional to the amount of the strain wave under the coil, on the output of the coil leads.

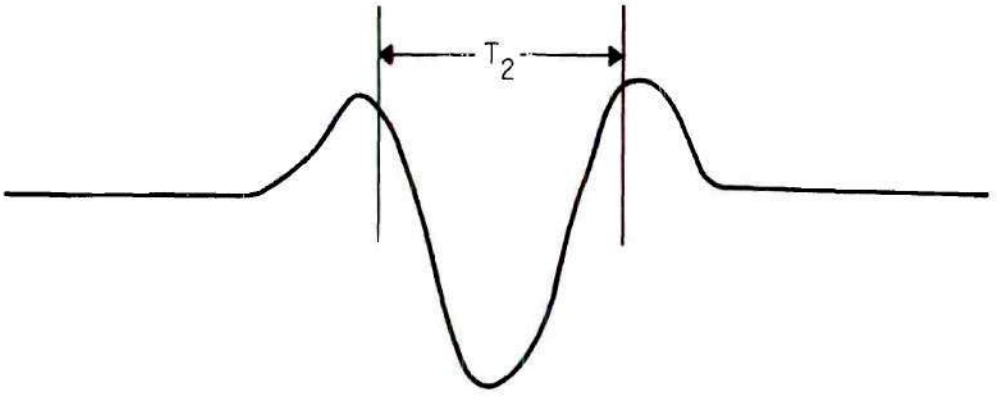
A variation of this configuration is shown in Figure 27(b) and is the type used in the buffer. Here the input pulses are converted to longitudinal strain waves, which in turn produce torsional strain, or shear mode strain, in the delay line. This mode is desireable because shear strain waves travel at 60 per cent of the velocity of longitudinal strain waves and suffer less dispersion in curved portions of the wire.[10] The shear strain waves are reconverted to longitudinal strain waves at the output end, and an output voltage pulse produced as before.

The delay line seriously attenuates and deforms input pulses, as may be seen in Figure 28(a) and (b). Typical voltage attenuation is on the order of 40 to 60 decibels; therefore special sense or reader circuits are needed on the output of the delay line.

The Delay Line Driver of the Memory and Address Logic Block is a NPN transistor switch which has the input transducer of the delay line connected in the collector circuit, as shown in Figure 29. A diode is



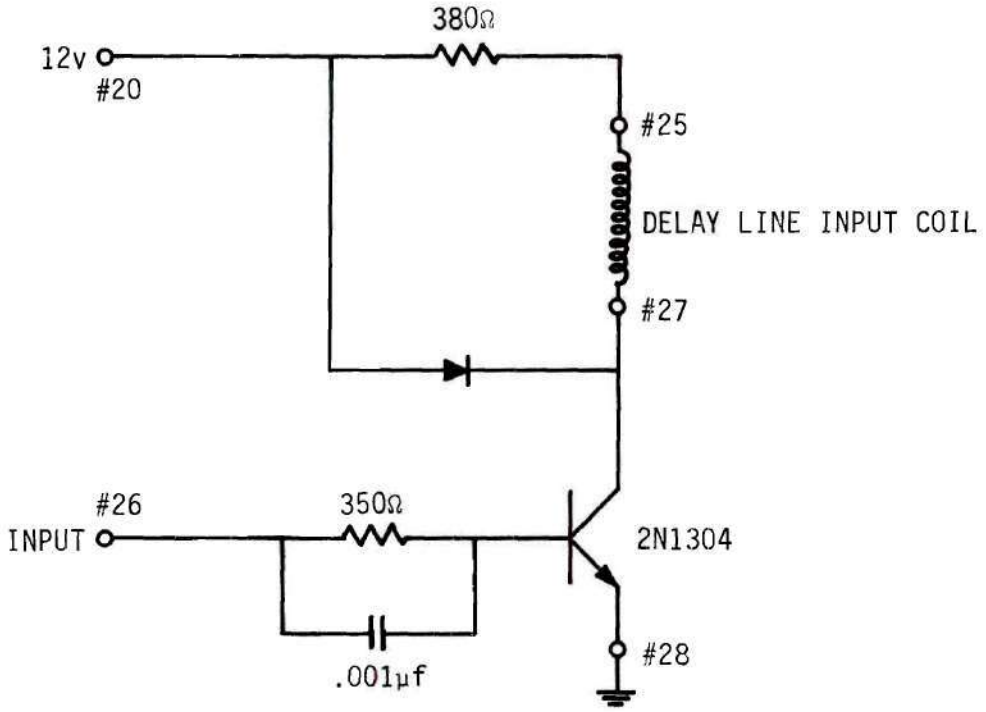
(a) INPUT WAVE FORM



(b) OUTPUT WAVE FORM

NOTE: $T_2 \simeq 2T_1$

Figure 28. Delay Line Input and Output Wave Forms.



NOTE: NUMBERS REFER TO PIN NUMBERS

Figure 29. Delay Line Driver.

placed in parallel with the transducer to limit inductive surges.

The Delay Line Reader is shown in Figure 30. This circuit consists of two amplifier stages and a Schmitt Trigger. The amplifiers raise the level of the output signal, which is typically 12 milli-volts maximum, to a level that may be detected by the Schmitt Trigger. The Schmitt Trigger is a voltage-sensitive circuit which produces a minus output level when a certain input level is reached. The magnitude of the input level trigger point may be adjusted by means of the 1000 ohm adjustable potentiometer. In practice, the potentiometer is adjusted until reliable operation of the memory system is obtained, as explained in the section on buffer adjustment.

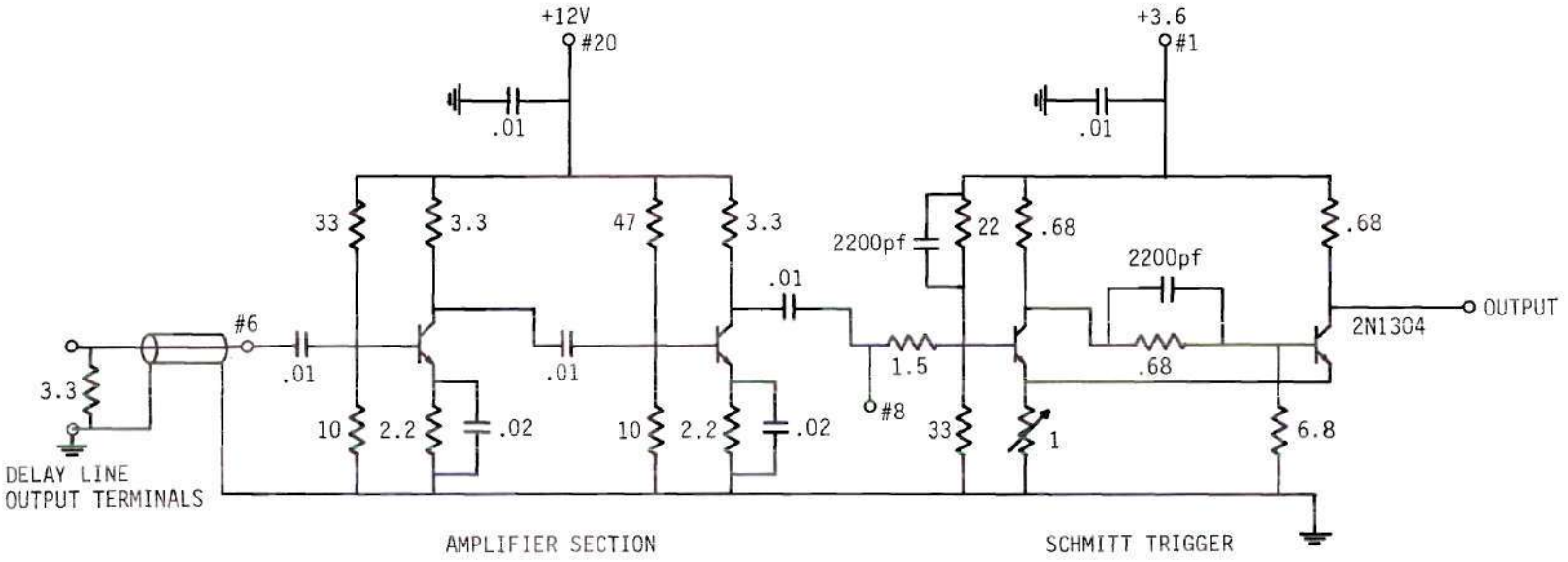
Stepping Motor

The stepping motor used was a bifilar, permanent-magnet type. The motor may be termed a two-phase synchronous motor with center-tapped field coils. The field is wound on the stator and is arranged in a number of poles; the rotor is a two-pole permanent magnet. A simplified representation of a four-pole motor is shown in Figure 31. When used as a synchronous motor, the field windings are connected to balanced, two-phase voltages, and the synchronous speed is given by

$$n = \frac{120f}{P} \text{ r.p.m.}$$

where f is the frequency, in cycles per second, of the applied voltage, and P is the number of poles.

To be used as a stepping motor the individual field coils may be energized in an orderly sequence, producing a stator field which



NOTE: ALL RESISTANCES IN KILOHMS
ALL CAPACITANCES IN μf UNLESS NOTED
NUMBERS REFER TO PIN NUMBERS

Figure 30. Delay Line Reader.

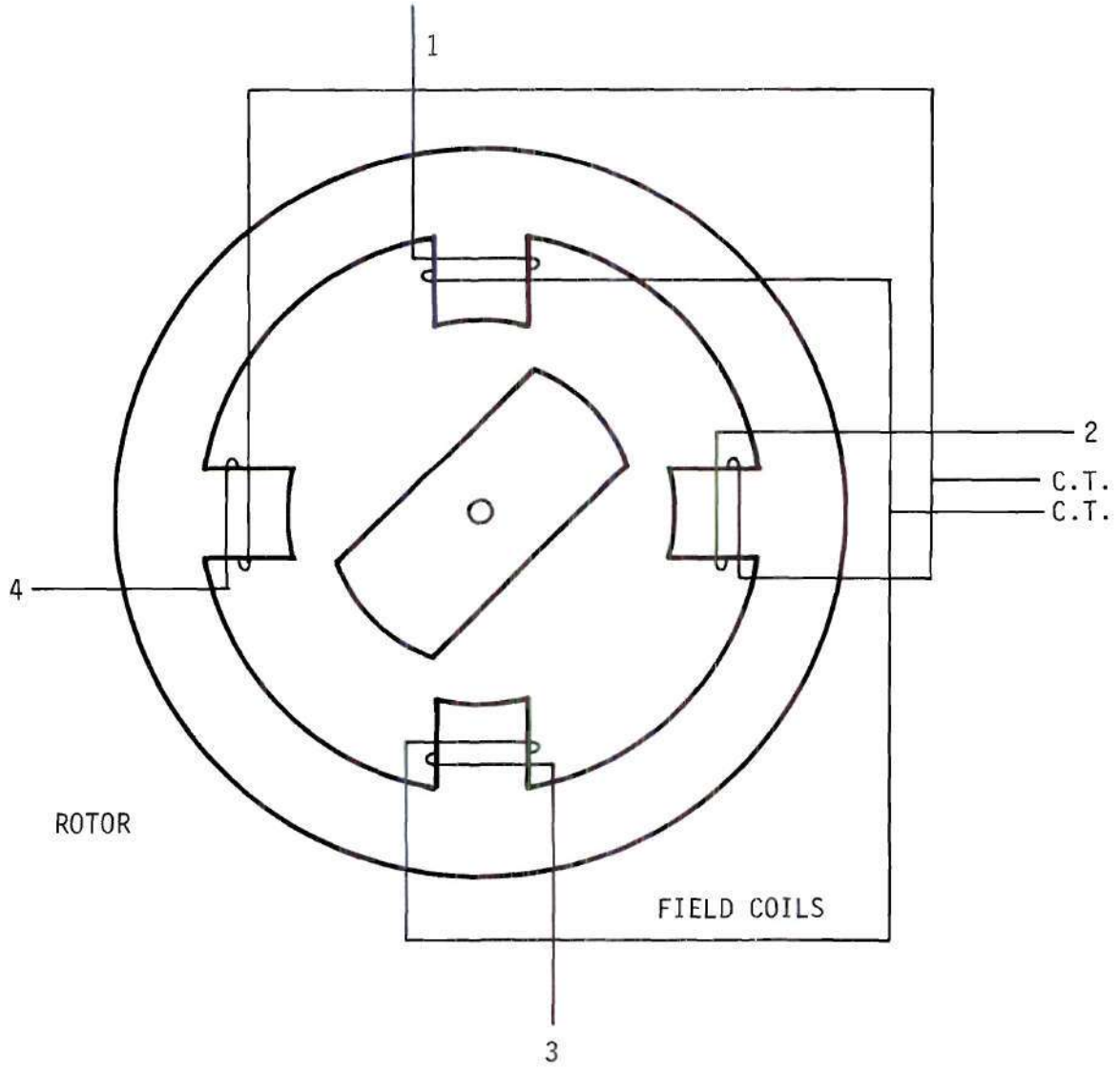


Figure 31. Four-pole Stepping Motor.

progresses in discrete increments of rotation and which the rotor follows as the rotor field aligns with the stator field. To provide uniform increments of rotation, the individual field coils can be energized in the order 1 2 3 4 1 . . ., giving clockwise rotation, or 1 4 3 2 1 . . ., giving counterclockwise rotation. To obtain more torque, due to a stronger stator field, adjacent field coils may be energized as the sequence 12, 23, 34, 41, 12, . . ., for clockwise rotation, and 12, 14, 43, 32, 12, . . ., for counterclockwise rotation. Each sequence gives a discrete rotation of 90 electrical degrees per pole pair.

The stepping motor used with the buffer was driven using the second method described above, as discussed in Chapter I. The motor used was specified to rotate at 72 r.p.m. when used as a synchronous motor at a line frequency of 60 c.p.s. Therfore,

$$P = \frac{120f}{n} = \frac{120 \times 60}{72} = 100 \text{ poles}$$

The 100 poles are arranged in groups of four, each group electrically the same as discussed above for the four-pole machine. To determine the number of mechanical degrees per sequence,

$$360 \text{ electrical degrees} = \frac{360 \text{ mechanical degrees}}{50 \text{ pole pairs}}$$

$$= 7.2 \text{ mechanical degrees;}$$

$$90 \text{ electrical degrees} = \frac{7.2 \text{ mechanical degrees}}{4}$$

$$= 1.8 \text{ mechanical degrees.}$$

The motor used was connected so that when the sign bit of an

input number was one, it rotated in a clockwise direction, and rotated counterclockwise for a zero sign bit when viewed from the shaft end.

Buffer Clock

The buffer clock was built as shown in the circuit diagram of Figure 32. The circuit is basically a crystal-controlled oscillator and amplifier which is driven so that switching of the output waveform results. The frequency of the crystal used was 506.500 KC ± .005 per cent, giving the very stable frequency needed for proper buffer operation. The output of the clock is then connected to several buffers so that driving capability is provided.

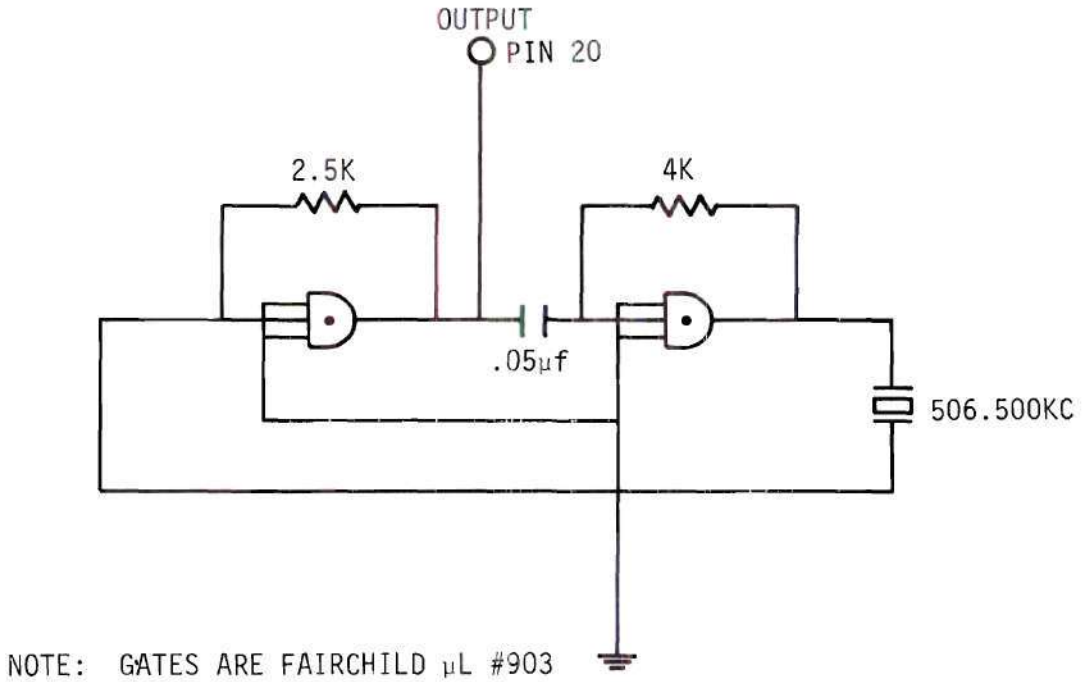


Figure 32. Buffer Clock.

APPENDIX II

TEST DATA

Equipment Used

1. Power Supply P_D: Electronic Research Associates Model 110DMC
2. Power Supply P_m: Sorenson Nobatron
3. Power Supply P_B: Sorenson Nobatron
4. Oscilloscope: TEKTRONIX Type 544 Plugin unit type 1A2

Extended Accuracy Test

Run Number	Position at Start	Position at End	Rotations CCW	Rotations CW	Run Number	Position at Start	Position at End	Rotations CCW	Rotations CW
1	99°	99°	5	5	39	99°	99°	5	5
2	99°	99°	5	5	40	99°	99°	5	5
3	99°	99°	5	5	41	99°	99°	5	5
4	99°	99°	5	5	42	99°	99°	5	5
5	99°	99°	5	5	43	99°	99°	5	5
6	99°	99°	5	5	44	99°	99°	5	5
7	99°	99°	5	5	45	99°	99°	5	5
8	99°	99°	5	5	46	99°	99°	5	5
9	99°	99°	5	5	47	99°	99°	5	5
10	99°	99°	5	5	48	99°	99°	5	5
11	99°	99°	5	5	49	99°	99°	5	5
12	99°	99°	5	5	50	99°	99°	5	5
13	99°	99°	5	5	51	99°	99°	5	5
14	99°	99°	5	5	52	99°	99°	5	5
15	99°	99°	5	5	53	99°	99°	5	5
16	99°	99°	5	5	54	99°	99°	5	5
17	99°	99°	5	5	55	99°	99°	5	5
18	99°	99°	5	5	56	99°	99°	5	5
19	99°	99°	5	5	57	99°	99°	5	5
20	99°	99°	5	5	58	99°	99°	5	5
21	99°	99°	5	5	59	99°	99°	5	5
22	99°	99°	5	5	60	99°	99°	5	5
23	99°	99°	5	5	61	99°	99°	5	5
24	99°	99°	5	5	62	99°	99°	5	5
25	99°	99°	5	5	63	99°	99°	5	5
26	99°	99°	5	5	64	99°	99°	5	5
27	99°	99°	5	5	65	99°	99°	5	5
28	99°	99°	5	5	66	99°	99°	5	5
29	99°	99°	5	5	67	99°	99°	5	5
30	99°	99°	5	5	68	99°	99°	5	5
31	99°	99°	5	5	69	99°	99°	5	5
32	99°	99°	5	5	70	99°	99°	5	5
33	99°	99°	5	5	71	99°	99°	5	5
34	99°	99°	5	5	72	99°	99°	5	5
35	99°	99°	5	5	73	99°	99°	5	5
36	99°	99°	5	5	74	99°	99°	5	5
37	99°	99°	5	5	75	99°	99°	5	5
38	99°	99°	5	5	76	99°	99°	5	5

Extended Accuracy Test Continued

Run Number	Position at Start	Position at End	Rotations CCW	Rotations CW	Run Number	Position at Start	Position at End	Rotations CCW	Rotations CW
77	99°	99°	5	5	121	270°	270°	5	5
78	99°	99°	5	5	122	270°	270°	5	5
79	99°	99°	5	5	123	270°	270°	5	5
80	99°	99°	5	5	124	270°	270°	5	5
81	99°	99°	5	5	125	270°	270°	5	5
82	99°	99°	5	5	126	270°	270°	5	5
83	99°	99°	5	5	127	270°	270°	5	5
84	99°	99°	5	5	128	270°	270°	5	5
85	99°	99°	5	5	129	270°	270°	5	5
86	99°	99°	5	5	130	270°	270°	5	5
87	99°	99°	5	5	131	270°	270°	5	5
88	99°	99°	5	5	132	270°	270°	5	5
89	99°	99°	5	5	133	270°	270°	5	5
90	99°	99°	5	5	134	270°	270°	5	5
91	99°	99°	5	5	135	270°	270°	5	5
92	99°	99°	5	5	136	270°	270°	5	5
93	99°	99°	5	5	137	270°	270°	5	5
94	99°	99°	5	5	138	270°	270°	5	5
95	99°	99°	5	5	139	270°	270°	5	5
96	99°	99°	5	5	140	270°	270°	5	5
97	99°	99°	5	5	141	270°	270°	5	5
98	99°	99°	5	5	142	270°	270°	5	5
99	99°	99°	5	5	143	270°	270°	5	5
100	99°	99°	5	5	144	270°	270°	5	5
101	270°	270°	5	5	145	270°	270°	5	5
102	270°	270°	5	5	146	270°	270°	5	5
103	270°	270°	5	5	147	270°	270°	5	5
104	270°	270°	5	5	148	270°	270°	5	5
105	270°	270°	5	5	149	270°	270°	5	5
106	270°	270°	5	5	150	270°	270°	5	5
107	270°	270°	5	5	151	270°	270°	5	5
108	270°	270°	5	5	152	270°	270°	5	5
109	270°	270°	5	5	153	270°	270°	5	5
110	270°	270°	5	5	154	270°	270°	5	5
111	270°	270°	5	5	155	270°	270°	5	5
112	270°	270°	5	5	156	270°	270°	5	5
113	270°	270°	5	5	157	270°	270°	5	5
114	270°	270°	5	5	158	270°	270°	5	5
115	270°	270°	5	5	159	270°	270°	5	5
116	270°	270°	5	5	160	270°	270°	5	5
117	270°	270°	5	5	161	270°	270°	5	5
118	270°	270°	5	5	162	270°	270°	5	5
119	270°	270°	5	5	163	270°	270°	5	5
120	270°	270°	5	5	164	270°	270°	5	5

Extended Accuracy Test Continued

Run Number	Position at Start	Position at End	Rotations CCW	Rotations CW	Run Number	Position at Start	Position at End	Rotations CCW	Rotations CW
165	270°	270°	5	5	184	270°	270°	5	5
166	270°	270°	5	5	185	270°	270°	5	5
167	270°	270°	5	5	186	270°	270°	5	5
168	270°	270°	5	5	187	270°	270°	5	5
169	270°	270°	5	5	188	270°	270°	5	5
170	270°	270°	5	5	189	270°	270°	5	5
171	270°	270°	5	5	190	270°	270°	5	5
172	270°	270°	5	5	191	270°	270°	5	5
173	270°	270°	5	5	192	270°	270°	5	5
174	270°	270°	5	5	193	270°	270°	5	5
175	270°	270°	5	5	194	270°	270°	5	5
176	270°	270°	5	5	195	270°	270°	5	5
177	270°	270°	5	5	196	270°	270°	5	5
178	270°	270°	5	5	197	270°	270°	5	5
179	270°	270°	5	5	198	270°	270°	5	5
180	270°	270°	5	5	199	270°	270°	5	5
181	270°	270°	5	5	200	270°	270°	5	5
182	270°	270°	5	5					
183	270°	270°	5	5					

RANDOM ACCURACY TEST

Sign Bit = 0
Degrees to nearest .5°

Run Number	Number Set In	Position at Start, Degrees	Position at End CCW Revolutions	Degrees	Total Degrees	No. Steps = Deg./1.8°	Line Clear
1	0	175	0	175	0	0	yes
2	1	175	0	176.5	1.5	1	yes
3	2	176.5	0	180	3.5	2	yes
4	3	180	0	186	6	3	yes
5	4	186	0	193	7	4	yes
6	5	193	0	202	9	5	yes
7	6	202	0	212.5	10.5	6	yes
8	7	212.5	0	225	12.5	7	yes
9	8	225	0	240	15	8	yes
10	9	240	0	256	16	9	yes
11	10	256	0	274	18	10	yes
12	20	274	0	310	36	20	yes
13	30	310	0	3.5	53.5	30	yes
14	40	3.5	0	75.5	72	40	yes
15	50	75.5	0	166	89.5	50	yes
16	60	166	0	273.5	107.5	60	yes
17	70	273.5	0	39	125.5	70	yes
18	80	39	0	183.5	144.5	80	yes
19	90	183.5	0	345.5	162	90	yes
20	100	345.5	0	166	180.5	100	yes
21	200	166	1	166	360	200	yes
22	300	166	1	345.5	539.5	300	yes
23	400	345.5	2	345.5	720	400	yes
24	500	345.5	2	166	900.5	500	yes
25	600	166	3	166	1080	600	yes
26	700	166	3	345.5	1260.5	700	yes
27	800	345.5	4	345.5	1440	800	yes
28	900	345.5	4	166	1620.5	900	yes
29	1000	166	5	166	1800	1000	yes
30	1023	166	5	207.5	1841.5	1023	yes
31	72	207.5	0	336.5	129	72	yes
32	165	336.5	0	273.5	297.0	165	yes
33	239	273.5	1	343.5	430	239	yes

RANDOM ACCURACY TEST CONTINUED

Sign Bit = 0
Degrees to Nearest.5°

Run Number	Number Set In	Position at Start, Degrees	Position at End CCW Revolution	Degrees	Total Degrees	No. Steps = $\frac{\text{Deg}}{1.8°}$	Line Clear
34	304	343.5	1	171.5	548	304	yes
35	420	171.5	2	207	755.5	420	yes
36	589	207	2	187.5	1060.5	589	yes
37	628	187.5	3	238.0	1130.5	628	yes
38	791	238.0	3	222	1424	791	yes
39	1007	199	5	131.5	1812.5	1007	yes

RANDOM ACCURACY TEST

Sign Bit = 1
Degrees to nearest.5°

Run Number	Number Set In	Position at Start, Degrees	Position at End Revolution CW	Degrees	Total Degrees	No. Steps = Deg./1.8°	Line Clear
1	0	99	0	99	0	0	yes
2	1	99	0	101	2	1	yes
3	2	101	0	104.5	3.5	2	yes
4	3	104.5	0	110	5.5	3	yes
5	4	110	0	117	7	4	yes
6	5	117	0	126	9	5	yes
7	6	126	0	137	11	6	yes
8	7	137	0	149.5	12.5	7	yes
9	8	149.5	0	164.0	14.5	8	yes
10	9	164.0	0	180	16	9	yes
11	10	180	0	198.5	18.5	10	yes
12	20	198.5	0	234	35.5	20	yes
13	30	234	0	288	54	30	yes
14	40	288	0	0	72	40	yes
15	50	0	0	90	90	50	yes
16	60	90	0	198	108	60	yes
17	70	198	0	324	126	70	yes
18	80	324	0	108	144	80	yes
19	90	108	0	270	162	90	yes
20	100	270	0	90	180	100	yes
21	200	90	1	90	360	200	yes
22	300	90	1	270	540	300	yes
23	400	270	2	270	720	400	yes
24	500	270	2	90	900	500	yes
25	600	90	3	90	1080	600	yes
26	700	90	3	270	1260	700	yes
27	800	270	4	270	1440	800	yes
28	900	270	4	90	1620	900	yes
29	1000	90	5	90	1800	1000	yes
30	1023	90	5	131.5	1841.5	1023	yes
31	86	131.5	0	286	154.5	86	yes
32	125	286	0	151.5	225.5	125	yes
33	232	151.5	1	209	417.5	232	yes

RANDOM ACCURACY TEST CONTINUED

Sign Bit = 1
Degrees to nearest 5°

Run Number	Number Set In	Position at Start, Degrees	Position at End Revolution CW	Degrees	Total Degrees	No. Steps = Deg./1.8°	Line Clear
34	347	209	1	113.5	624.5	347	yes
35	598	113.5	2	110	1076.5	598	yes
36	619	110	3	144	1114	619	yes
37	763	144	3	77.5	1373.5	763	yes
38	854	77.5	4	175.0	1537.5	854	yes
39	905	175.0	4	3.5	1628.5	905	yes

ADDRESS ACCURACY TEST

Number
Set In = 1000

Run Number	Address Set In	Chosen Address Observed To Decrement	Motor Observed to Step	Sign Bit	Sign Bit
1	1	yes	no	1	0
2	2	yes	no	1	0
3	3	yes	no	1	0
4	4	yes	no	1	0
5	5	yes	no	1	0
6	6	yes	no	1	0
7	7	yes	no	1	0
8	8	yes	no	1	0
9	9	yes	no	1	0
10	10	yes	no	1	0
11	11	yes	no	1	0
12	12	yes	no	1	0
13	13	yes	no	1	0
14	14	yes	no	1	0
15	15	yes	no	1	0
16	16	yes	no	1	0
17	17	yes	no	1	0
18	18	yes	no	1	0
19	19	yes	no	1	0
20	20	yes	no	1	0
21	21	yes	no	1	0
22	22	yes	no	1	0
23	23	yes	no	1	0
24	24	yes	no	1	0
25	25	yes	no	1	0
26	--	---	--	-	-
27	27	yes	no	1	0
28	28	yes	no	1	0
29	29	yes	no	1	0
30	30	yes	no	1	0
31	31	yes	no	1	0
32	32	yes	no	1	0
33	33	yes	no	1	0
34	34	yes	no	1	0
35	35	yes	no	1	0

ADDRESS ACCURACY TEST CONTINUED

Number Set In = 1000

Run Number	Address Set In	Chosen Address Observed To Decrement	Motor Observed to Step	Sign Bit	Sign Bit
36	36	yes	no	1	0
37	37	yes	no	1	0
38	38	yes	no	1	0
39	39	yes	no	1	0
40	40	yes	no	1	0
41	41	yes	no	1	0
42	42	yes	no	1	0
43	43	yes	no	1	0
44	44	yes	no	1	0
45	45	yes	no	1	0
46	46	yes	no	1	0
47	47	yes	no	1	0
48	48	yes	no	1	0
49	49	yes	no	1	0
50	50	yes	no	1	0
51	51	yes	no	1	0
52	52	yes	no	1	0
53	53	yes	no	1	0
54	54	yes	no	1	0
55	55	yes	no	1	0
56	56	yes	no	1	0
57	57	yes	no	1	0
58	58	yes	no	1	0
59	59	yes	no	1	0
60	60	yes	no	1	0
61	61	yes	no	1	0
62	62	yes	no	1	0
63	63	yes	no	1	0

UPDATE TEST

± 1000

Run Number	1st Number Set In	Sign Bit	2nd Number Set In	Sign Bit	Position at Start, Degrees	Position at End, Degrees	Net Steps
1	1000	0	1000	1	160	160	0
2	1000	1	1000	0	160	160	0
3	1000	0	1000	1	160	160	0
4	1000	1	1000	0	160	160	0
5	1000	0	1000	1	160	160	0
6	1000	1	1000	0	160	160	0
7	1000	0	1000	1	160	160	0
8	1000	1	1000	0	160	160	0
9	1000	0	1000	1	160	160	0
10	1000	1	1000	0	160	160	0
11	1000	0	1000	1	160	160	0
12	1000	1	1000	0	160	160	0
13	1000	0	1000	1	160	160	0
14	1000	1	1000	0	160	160	0
15	1000	0	1000	1	160	160	0
16	1000	1	1000	0	160	160	0
17	1000	0	1000	1	160	160	0
18	1000	1	1000	0	160	160	0
19	1000	0	1000	1	160	160	0
20	1000	1	1000	0	160	160	0
21	1000	0	1000	1	160	160	0
22	1000	1	1000	0	160	160	0
23	1000	0	1000	1	160	160	0
24	1000	1	1000	0	160	160	0
25	1000	0	1000	1	160	160	0

UPDATE TEST

500 + 500

Run Number	1st Number Set In	Sign Bit	2nd Number Set In	Sign Bit	Position at Start, Degrees	Position at End Revolutions CCW	Position at End Degrees	Net Steps
1	500	0	500	0	270	5	270	1000
2	500	0	500	0	270	5	270	1000
3	500	0	500	0	270	5	270	1000
4	500	0	500	0	270	5	270	1000
5	500	0	500	0	270	5	270	1000
6	500	0	500	0	270	5	270	1000
7	500	0	500	0	270	5	270	1000
8	500	0	500	0	270	5	270	1000
9	500	0	500	0	270	5	270	1000
10	500	0	500	0	270	5	270	1000
11	500	0	500	0	270	5	270	1000
12	500	0	500	0	270	5	270	1000
13	500	0	500	0	270	5	270	1000
14	500	0	500	0	270	5	270	1000
15	500	0	500	0	270	5	270	1000
16	500	0	500	0	270	5	270	1000
17	500	0	500	0	270	5	270	1000
18	500	0	500	0	270	5	270	1000
19	500	0	500	0	270	5	270	1000
20	500	0	500	0	270	5	270	1000
21	500	0	500	0	270	5	270	1000
22	500	0	500	0	270	5	270	1000
23	500	0	500	0	270	5	270	1000
24	500	0	500	0	270	5	270	1000
25	500	0	500	0	270	5	270	1000

UPDATE TEST

500 + 500

Run Number	1st Number Set In	Sign Bit	2nd Number Set In	Sign Bit	Position at Start, Degrees	Position at End Revolutions CW	Position at End Degrees	Net Steps
1	500	1	500	1	270	5	270	1000
2	500	1	500	1	270	5	270	1000
3	500	1	500	1	270	5	270	1000
4	500	1	500	1	270	5	270	1000
5	500	1	500	1	270	5	270	1000
6	500	1	500	1	270	5	270	1000
7	500	1	500	1	270	5	270	1000
8	500	1	500	1	270	5	270	1000
9	500	1	500	1	270	5	270	1000
10	500	1	500	1	270	5	270	1000
11	500	1	500	1	270	5	270	1000
12	500	1	500	1	270	5	270	1000
13	500	1	500	1	270	5	270	1000
14	500	1	500	1	270	5	270	1000
15	500	1	500	1	270	5	270	1000
16	500	1	500	1	270	5	270	1000
17	500	1	500	1	270	5	270	1000
18	500	1	500	1	270	5	270	1000
19	500	1	500	1	270	5	270	1000
20	500	1	500	1	270	5	270	1000
21	500	1	500	1	270	5	270	1000
22	500	1	500	1	270	5	270	1000
23	500	1	500	1	270	5	270	1000
24	500	1	500	1	270	5	270	1000
25	500	1	500	1	270	5	270	1000

APPENDIX III

Parts List and Specifications

Included here is the parts list of major items used, and outline specifications for the logic, motor, and delay line.

PARTS LIST

Item	Nomenclature	Manufacturer	Part or Type Number	Number Used
1.	2-NOR	S.E.L.[1]	8522	59
2.	3 J-K Flip-Flop	S.E.L.	8528	48
3.	Buffer	S.E.L.	8501	15
4.	Inverter	S.E.L.	8525	12
5.	2-NOR Gate	S.E.L.	8527	17
6.	3-NOR Expender	S.E.L.	8531	17
7.	Synchronous Motor	Superior Electric	5550	1
8.	Delay Line	C.D.C.	MT762-2022	1
9.	NPN Transistor	R.C.A.	40250	4
10.	NPN Transistor	Amperex	2N1304	5
11.	3-NOR	Fairchild	μ L #903	2
12.	Cabinet	S A	EC5A	1
13.	Chasis	Bud	AC426	1
14.	Panels	John Stevens Engraving Co.	N.A.	2
15.	Card Carrier	S.E.L.[2]	N.A.	1
16.	Crystal	Texas Crystals	N.A.	1

Note: S.E.L. = Systems Engineering Laboratory
C.D.C. = Computer Devices Corporation
R.C.A. = Radio Corporation of America
S.A. = Scientific Atlanta

Approximate cost: $1500

[1] The logic is Fairchild μL 900 series, card mounted by S.E.L.

[2] Card Carrier uses ELCO series 7022 connectors

COMPONENT SPECIFICATIONS

Item		
1.	SEL Logic Cards	V_c = 3.6 V, Logic is Fairchild Industries Line +15 degrees centigrade to +55 degrees centigrade
2.	Delay Line	Delay, 2000 μ sec. $\pm$ 2 μ sec. Temperature Drift (time), 0-50 degrees centigrade; $\pm$ 100 nanosecond.
3.	Synchronous Motor	V = 14 V, I = .53 ampere, 75 degree centigrade Class B Insulation. Torque = 50 ounce - Inch.

LIST OF REFERENCES

1. Schall, W.C., "Direct Digital Control - The First Two Years," Proceedings of the Fifth National Chemical and Petroleum Instrumentation Symposium, Wilmington Delaware, May 4-5, 1964, pp. 1-6.

2. "Report on Process Computer Control," Instrument Society of America Vol. 10, No. 6, January 1963, pp. 47-48.

3. Moreley, R.A and Cundal, C.M., "The Ferranti System and Experience with Direct Digital Control," IEEE International Convention Record, Part III. New York, March 1965, pp. 90-100.

4. Williams, Theodore J., "Process Control," Industrial and Engineering Chemistry, Vol. 57, No. 12, December 1965, pp. 33-43.

5. Karp, H.R., "Direct Digital Control Up to the Hilt," ISA Journal, Vol. 13, No. 2, February 1966, pp. 13-15.

6. Gallier, P.W., "Surveying Process Digital Control," IEEE International Convention Record, Part IV. New York, 1965, pp 2-9.

7. Williard, F.G., Kirk, G.J. Jr., Radcliffe, P.S., Schay, J.R., "A Method of Implementing Direct Digital Control," Instrumentation in the Chemical and Petroleum Industries, May 4-5, 1964, pp. 21-29.

8. Andelman, S. J., "Real-Time I/O Techniques to Reduce System Costs," Computer Design, Vol. 5, May 1966, pp. 48-54.

9. Elmer, H.G., "Sonic Delay Line used for Buffering of Pulse Inputs to a Control Computer System," 1964 Joint Automatic Control Conference, pp. 52-54.

10. Ellis, Dave, "Delay Line Memories," Data Storage Handbook. Pittsburg, Pennsylvania, Instruments Publishing Company, 1962, pp. 68-70.